INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

**Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA**

UMI[®]
800-521-0600

Alarm Correlation in ATM Networks

by

Chunhui Teng, B.Eng. & M.Eng.

A thesis submitted to

the Faculty of Graduate Studies and Research

in partial fulfillment of

the requirements for the degree of

Master of Engineering

Ottawa-Carleton Institute for Electrical Engineering
Faculty of Engineering
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario
March 1, 1999
© copyright
1999, Chunhui Teng

**National Library
of Canada**

**Acquisitions and
Bibliographic Services**

**395 Wellington Street
Ottawa ON K1A 0N4
Canada**

**Bibliothèque nationale
du Canada**

**Acquisitions et
services bibliographiques**

**395, rue Wellington
Ottawa ON K1A 0N4
Canada**

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-43348-X

Canada

The undersigned recommend to the Faculty of Graduate Studies and Research
acceptance of the thesis

ALARM CORRELATION IN ATM NETWORKS

submitted by Chunhui Teng, B.Eng. & M.Eng.

in partial fulfillment of
the requirements for the degree of
Master of Engineering

Thesis Supervisor

Chair, Department of Systems and Computer Engineering

Carleton University

May 25, 1999

Abstract

In ATM networks, a large number of alarms exist to signal any abnormal behavior of the networks. As network faults typically result in a great number of alarms, correlating these different alarms and identifying their source is a major problem in fault management of ATM networks. This thesis discusses alarm correlation in ATM networks, and application of an alarm correlation algorithm to alarm correlation in ATM networks. We describe important faults/alarms in ATM networks and the way they propagate in ATM networks, and how to do fault localization and identification.

Acknowledgements

I would like to express my thanks to my thesis supervisor, Professor Bernard Pagurek, for his guidance throughout my graduate studies. I would also like to thank my manager, Juan Arroyo, for his support of thesis research work.

At last, I thank my wife, Xiaobo, my daughter, Yue, my son, Fei, for their support, patience, and love.

Contents

Chapter 1 Introduction	1
1.1 The Goals of Network Management	1
1.2 The Requirements of Network Management	2
1.2.1 From Planning/Deploying personnel	2
1.2.2 From surveillance/maintenance personnel	3
1.2.3 From personnel providing services via ATM	3
1.3 The Characteristics of ATM Networks and its Impact on Network Management	3
1.3.1 How Connection-Oriented Networks Impact Management	4
1.3.2 Media Characteristics	5
1.3.3 Complexity of ATM Networks	6
1.3.4 QOS Requirements	6
1.3.5 Real-time Traffic	6
1.3.6 Volume of Alarms	6
1.3.7 Built-in manageability - OAM cells	7
1.3.8 Problems of conventional approaches	7
1.3.9 Other issues	7
1.4 Work Done for the Thesis	7
Chapter 2 The Standards for Management of ATM Network Elements.....	10
2.1 The IETF SNMP MIBs	10
2.1.1 MIB-II (Interfaces for physical Layer, Cell Layer, AAL3/4, and AAL5)	11
2.1.2 SONET/SDH MIB	12
2.1.3 ATM MIB (RFC2515)	12
2.1.4 ATM Supplemental MIB	13
2.1.5 Accounting MIB	15
2.1.6 Performance History MIB	15
2.1.7 ATM Test MIB	16
2.2 The ATM Forum SNMP MIBs	16
2.2.1 ILMI MIB	16
Link Management MIB	17
ATM Address Registration MIB	17
2.2.2 ATM RMON MIB	18
2.3 Customer Network Management (CNM) and M4 Management Interface	19
2.3.1 Approved specifications related to the model	21
2.3.2 CNM M3 Specification	21
2.3.3 SNMP M4 Network Element View MIB	21
2.4 What Additional Information Would be Important	22

Chapter 3 Aspects of Management of ATM Networks.....	23
3.1 Fault Management	23
3.1.1 Alarm surveillance and correlation	23
Physical layer	23
ATM layer	24
3.1.2 Fault localization and testing	25
3.1.3 Alarm correlation	25
3.2 Performance Management	25
3.2.1 Performance monitoring	25
Physical layer	25
ATM layer	26
Upper Layers	26
3.2.2 Protocol performance monitoring and test	26
Physical Layer	27
ATM layer	28
AAL Layer	29
Upper Layers	30
3.2.3 Network traffic monitoring	31
3.3 Configuration Management	31
3.4 Accounting Management	32
3.5 Security Management	32
3.6 Practical experience in managing ATM LANs	32
3.6.1 Work done related to ATM network management	32
3.6.2 The topology of the networks	33
3.6.3 Configuration Management	34
NM Console configuration management	35
Real time configuration reporting	36
Non-real time configuration reporting, graphing	36
3.6.4 Performance Management	38
MIB objects related to performance management	38
Event and alarm generation (performance thresholding)	40
Performance reporting and graphing	40
Performance data collection/reduction and reporting/graphing	42
3.6.5 Fault Management	42

Chapter 4 Alarm Correlation Algorithms	43
4.1 Introduction	43
4.2 Katzela-Schwartz approach	43
4.2.1 Definition of the problem and model	43
4.2.2 Fault identification	49
4.2.3 Exhaustive search	52
4.2.4 Greedy search	52
4.2.5 Divide and Conquer	53
Input	55
Output	56
Method	56

Example	57
Performance Analysis	59
4.3 Brunet's algorithm (Temporal model)	62
4.3.1 Definition of the problem and model	62
4.3.2 Algorithm	68
4.4 Comparisons of alarm correlation algorithms	78
4.4.1 Scalability	79
4.4.2 Evaluation of Brunet's algorithm	79
4.4.3 Performance improvements to Brunet's algorithm	82
4.4.4 Comparisons	84
4.5 Shortcomings	85
Chapter 5 Alarm Correlation in ATM Networks	87
5.1 Alarms and their propagation in ATM networks	87
5.1.1 Model of ATM Networks	87
5.1.2 Important Alarm/Error Propagation	88
Alarm propagation between switches	95
Alarm propagation between ports in a switch	97
Combined causal model	99
Probabilities	103
5.1.3 The sources of the alarms	105
5.2 Application of Brunet's algorithm	109
5.2.1 Extension of the causal model	109
5.2.2 Generation of the causal graphs	114
5.2.3 Performance measurements	123
Chapter 6 Conclusions	125
6.1 Work done for this thesis	126
6.2 Future work	127

Figures

Figure 2.1 ATM Forum's Management Model	20
Figure 3.1 Topology of the networks	34
Figure 3.2 A sample ad hoc configuration report	36
Figure 3.3 Sample daily configuration report	37
Figure 3.4 Sample mperl script	41
Figure 3.5 Sample output from mperl script	41
Figure 4.1 (a) Network Example (b) Dependency Graph	46
Figure 4.2 Alarm Clusters	48
Figure 4.3 (a) Dependency Graph (b) Alarm Clusters	58
Figure 4.4 Phase I of the Maximum Mutual Dependency Algorithm	60
Figure 4.5 Phase II of the Maximum Mutual Dependency Algorithm	61
Figure 4.6 (a) Directed tree and (b) reversed tree	65
Figure 4.7 Cause-Alarm graph	73

Figure 5.1	The model of ATM networks.....	88
Figure 5.2	Alarm propagation in ATM networks (Physical Layer)	90
Figure 5.3	Alarm propagation in ATM networks (ATM layer)	91
Figure 5.4	Alarm propagation in ATM networks (AAL1).....	92
Figure 5.5	Alarm propagation in ATM networks (AAL3/4).....	93
Figure 5.6	Alarm propagation in ATM networks (AAL5).....	94
Figure 5.7	Alarm propagation in ATM networks (SAAL)	95
Figure 5.8	Important problems propagated between connected ports of two switches 96	
Figure 5.9	An example of alarm propagation between neighboring switches... 97	
Figure 5.10	Alarm propagation between ports in the same switch	98
Figure 5.11	An example of alarm propagation between ports in a switch.....	99
Figure 5.12	A sample causal model.....	102
Figure 5.13	The causal model generator	115
Figure 5.14	The sample network topology description file.....	116
Figure 5.15	The sample network.....	117
Figure 5.16	The sample topology specific causal model	119
Figure 5.17	The sample alarm file.....	121
Figure 5.18	The sample output of the alarm correlation program.....	122

Tables

Table 2.1	RFCs evaluated by the IETF AToM MIB Working Group	11
Table 2.2	Internet-Drafts edited by the IETF AToM MIB Working Group	11
Table 2.3	Functions supported by the AToM MIB	12
Table 2.4	Functions supported by the ATM Supplemental MIB	14
Table 2.5	Functions supported by the ILMi MIB	17
Table 2.6	Functions supported by the ATM Address Registration MIB	18
Table 2.7	Approved specifications related to CNM based on SNMP	21
Table 4.1	Comparisons of alarm correlation algorithms	85
Table 5.1	The mappings between the first digit of vertices and switches/ports	100
Table 5.2	The mappings between the last two digits of vertices and faults/alarms	100
Table 5.3	The sources of alarms	105
Table 5.4	Alarms caused by the fault "Traffic disagreement or wrong policing" at PortA of switch2	120
Table 5.5	Related numbers	123

Glossary and Acronyms

AAL	ATM adaptation layer
ABR	Available Bit Rate
AIS	Alarm Indication Signal
ATM	Asynchronous Transfer Mode
BICI	Broadband Inter-Carrier Interface
BIP	Bit Interleaved Parity
CBR	Constant Bit Rate
CLP	Cell Loss Priority
CNM	Custom Network Management
CPI	Common Part Indicator
ES	Errored Seconds
FEBE	Far End Block Error
FERF	Far End Receive Failure
HCS	Header Checking Sequence
HEC	Header Error Check
IETF	Internet Engineering Task Force
ILMI	(Interim) Integrated Layer Management Interface
IME	ILMI Management Entity
LAN	Local Area Network
LOF	Loss of Frame
LOP	Loss of Pointer
LOS	Loss of Signal
MAN	Metropolitan Area Network
MIB	Management Information Base
MID	ATM
NMS	Network Management System
OAM	Operation, Administration, and Management
OCD	Out of Cell Delineation
PLCP	Physical Layer Control Protocol
PNNI	Public Network-to-Network Interface
PT	Payload Type
PVC	Permanent Virtual Connection
PVCL	Permanent Virtual Channel Link

PVPL	Permanent Virtual Path Link
QoS	Quality of Service
RAI	Remote Alarm Indication
RDI	Remote Defect Indication
RFC	Request for Comments
SAAL	Signalling AAL
SAR	Segmentation and Reassembly
SDH	Synchronous Data Hierarchy
SDT	Structured Data Transfer
SDU	Service Data Unit
SEFS	Severely Errored Frame Seconds
SES	Severely Errored Seconds
SONET	Synchronous Optical Network
SVC	Switched Virtual Connection
TC	Transmission Convergence
UAS	Unavailable Seconds
UBR	Unspecified Bit Rate
UNI	User-Network Interface
UPC	Usage Parameter Control
VBR	Variable Bit Rate
VCC	Virtual Channel Connection
VCI	Virtual Channel Identifier
VCL	Virtual Channel Link
VPC	Virtual Path Connection
VPI	Virtual Path Identifier
VPL	Virtual Path Link
WAN	Wide Area Network

Chapter 1 Introduction

In this chapter, we will review the goals and requirements of network management and how characteristics of ATM networks impact the network management. Then we described briefly what we have done for the thesis.

1.1 The Goals of Network Management

The goals of corporate network management, as Minoli and Golway [Minoli1996] pointed out, are:

- High availability: The network-caused downtime should be very close to 0.
- Reliable/Serviceable
- Seamless: The networking devices from different vendors should be interoperable. Thus, operations and management procedures must be improved greatly in order to approach seamlessness with any technology, including ATM.
- High performance: The network managers must perform good evaluation of network performance and then react to any anomalies promptly.
- Responsive: The mean time to stabilize a complex network should be reduced.

- **Low system costs and operating costs:** The network planners and managers must sustain a good network performance under a minimum cost, including system costs (capital costs) and operating costs.
- **Improved feedback and diagnosis (event correlation):** Chargebacks and other procedures that allocate network costs to divisions and profit centers according to usage or other criteria are a fundamental dimension of network accounting. Prompt and accurate diagnosis, which depends on effective event correlation, is key to achieving high availability, high performance, and responsiveness.
- **Integrated proactive modeling:** Whenever possible, changes should be modeled, and their consequences evaluated, before they are put into practice.
- **Flexibility:** The network managers must accommodate different needs from different users.

1.2 The Requirements of Network Management

The network management has to assist various staff in the corporation. They have different requirements as enumerated below.

1.2.1 From Planning/Deploying personnel

The network planning and deploying people want to construct high performance networks, thus they need:

- Performance metrics for topology design and capacity planning.
- Traffic characteristics.
- Fault reports for redundancy design.

1.2.2 From surveillance/maintenance personnel

The surveillance/maintenance personnel require monitoring each layer to make sure switches and end systems are working properly and at an acceptable level of performance. They need effective and efficient alarm correlation to prevent not only significant misdirected efforts because of insufficient information, but also multiple corrective actions (possibly contradictory) because each alarm is handled independently. Alarm correlation for ATM networks is the area in which the main thesis effort has been placed.

1.2.3 From personnel providing services via ATM

The personnel providing services via ATM require monitoring QoS (quality of service) of connections. Some important connections can be monitored by details, e.g., cells transferred/dropped, cell delay and cell delay variation. But it is still necessary to make sure that all connections including SVCs (switched virtual circuits) are served well, i.e., the QoS requirements negotiated at connection setup time are met.

1.3 The Characteristics of ATM Networks and its Impact on Network Management

The ATM networks have their unique characteristics and they have great impact on network management. The characteristics are:

- Connection Oriented
- Media not shared and have high speed
- Complex

- QoS (quality of service) Requirements
- Real-time Traffic
- Volume of Information and Alarms
- Built-in manageability - OAM (operations, administrations, and maintenance) cells

The following sections will describe impact of each of these characteristics on network management.

1.3.1 How Connection-Oriented Networks Impact Management

In connectionless networks, monitored objects include network devices (routers, bridges, workstations, etc.) and links. While in ATM networks, in addition to network devices and links, VPC/VCCs (virtual path connections/virtual channel connections) are to be monitored. This increases complexity. The number of objects to be monitored may increase exponentially.

The number of VPC/VCCs may exclude the possibility that all VPC/VCCs be monitored. Thus it is important to have a way to evaluate the “health” of ATM networks by sampling some VPC/VCCs. We have to have criteria to choose VPC/VCCs to be monitored. For example, the VPC/VCCs can be chosen to be monitored based on their types (e.g. PVC, permanent virtual connections), owners, resource allocation, resource usages, and/or traversing networks with problems (congestion, high utilization).

In ATM networks, a connection has to be set up before information can be transferred. In the case of PVPC/PVCCs (Permanent Virtual Path Connection/Permanent Virtual Channel

Connection), connections need to be set up and torn down by a configuration manager. This calls for ability of switches to be set up by a commonly used protocol such as SNMP or CMIP so network operators can do this task easily, without needing to know which proprietary process should be followed to configure a switch from a vendor. The configuration management will have to be done more often than connectionless networks. Thus an efficient, easy-to-use configuration tool set is more desirable.

Virtual connections (PVC and SVC) can be across several switches and/or networks. So network management has to be distributed to permit access and achieve efficiency.

Some VPC/VCCs might not be fully (end-to-end, link by link) monitored because these networks may belong to different administration domains. There is still a long way to go before end-to-end management can be implemented. This is made possible by several M-interfaces defined by ATM Forum, which will be described in Chapter 2.

A new method to model networks is required, e.g., VPLs (virtual path links) are both sub-components of VPCs and switches. This requires changes to many current network management systems, which model networks as a “tree”, i.e., a network entity can not be a sub-component of two or more other network entities.

1.3.2 Media Characteristics

The media of ATM networks are not shared and usually have a high speed. This means that it is harder to employ a protocol analyzer like Sniffer for the Ethernet.

The media having a high speed means that an outage or fault is more costly. Thus it is more compelling that downtime be reduced as much as possible, and faults be resolved promptly.

1.3.3 Complexity of ATM Networks

ATM networks are very complex. This means that faults are more likely to be introduced if there is anything wrong. And in case of faults, performance degradations, or outages, it will take longer to resolve if there is no effective fault management tool.

1.3.4 QOS Requirements

Unlike other LAN/MAN/WAN, which have no guaranteed QoS, ATM networks have to offer QoS as negotiated at setup time.

ATM networks have to be kept working at high performance at all time because they have to provide guaranteed QoS. And any problem has to be resolved quickly.

1.3.5 Real-time Traffic

For non real-time traffic, long delays are a nuisance. For real-time traffic, long delays are not tolerated. So again, ATM networks have to be kept working at high level of performance at all time.

1.3.6 Volume of Alarms

Because VPC/VCCs depend on the lower layers, if there is any problem in the underlying layers, there could be alarm storms if the networks are not set up properly. This is the issue that we focus on in later chapters of this thesis.

1.3.7 Built-in manageability - OAM cells

ATM networks have OAM cells for layer management. Thus it would be desirable to integrate such capabilities in network management systems.

1.3.8 Problems of conventional approaches

Monitoring ATM networks can not be achieved by monitoring just the link utilization, error rate, and discards, which could be just sufficient for conventional networks. An ATM network/switch can have low utilization but can not meet the QOS requirements because of congestions caused by the bursty traffic.

1.3.9 Other issues

SVCs are short-lived and set-up by signalling outside management systems. It is hard for an NMS (network management system) to keep track of all SVCs.

1.4 Work Done for the Thesis

As we have discussed in the sections above, the complexity and importance of ATM networks demands proactive ATM network management. Alarm correlation is a key issue of ATM fault and performance management. The alarm correlation in ATM networks is very difficult and yet very important in ATM network management. However, there are almost no papers in the literature describing the alarm correlation in ATM networks; the only one [Lin1996] is just for congestion diagnosis.

The volume of ATM network alarms can be quite large, thus an efficient and effective algorithm is required to correlate alarms in ATM networks. However, the algorithms seen in the

literature, like Yemini's [Yemini1994] [Yemini1996] and Katzela's [Katzela1995], can not produce accurate diagnosis in many cases. Richard Brunet's thesis [Brunet1998] has addressed some of these problems, but there was no performance evaluation for the algorithm so the scalability is still unknown.

The following work has been done in this thesis to address these issues:

- Studied and described how faults/alarms propagate in ATM networks.
- Designed and developed a program to generate a causal graph for an ATM network automatically, based on the network topology.
- Evaluated and improved the scalability of Brunet's algorithm.
- Extended Brunet's model and applied Brunet's algorithm to do alarm correlation in ATM networks.
- Measured the performance of Brunet's algorithm and our improvements.

In this thesis, we will discuss ATM network management in general, which is the topic of Chapter 2 and Chapter 3, but concentrate on alarm correlation, which is the topic of Chapter 4 and 5.

This thesis is organized as follows. In chapter 2, we will first review some major MIB definitions from two standard bodies - ATM Forum and IETF, and usage of the MIB objects for monitoring and event reporting (most of alarms we list in later chapters to be correlated are based on these standards). Then we review the aspects of ATM network management and have

a brief introduction of how we have managed three ATM LANs in Nortel in Chapter 3. The alarm correlation algorithms and our improvements to Brunet's algorithm are reviewed in Chapter 4. Then we describe some important fault/error propagation in ATM networks, how to extend Brunet's model and how to apply Brunet's algorithm to do the alarm correlation, and the performance measurements of the algorithm in Chapter 5. Finally, we have conclusions and list future work in Chapter 6.

Chapter 2 The Standards for Management of ATM Network Elements

In this chapter we will have a quick review of various SNMP MIB modules from IETF working groups and ATM Forum and see how they can be used for five aspects of network management: configuration management, fault management, performance management, security management, and accounting management, which will be examined in more details in the next chapter. The MIB objects described in this chapter will be referred to in Chapter 5 when we discuss the origins of alarms in ATM networks.

2.1 The IETF SNMP MIBs

The IETF AToM MIB Working Group is evaluating several RFCs, including RFC 1695. The group also defines additional sets of managed objects, to reflect growing experience and industry requirements for management of ATM services, Frame based UNI (F-UNI), and ATM tests. The RFCs and current Internet-drafts produced by the group are shown in Table 2.1 and Table 2.2.

The subsequent sub-sections will review these Internet drafts and RFCs.

Table 2.1 RFCs evaluated by the IETF AToM MIB Working Group

RFC	Standard status
RFC 2515: Definitions of Managed Objects for ATM Management using SMIv2 (obsoletes RFC 1695)	Proposed Standard (Elective)
RFC 1595: Definitions of Managed Objects for the SONET/SDH Interface Type	Same as above
RFC 2493: Textual Conventions for MIB Modules Using Performance History Based on 15 Minute Interval	Same as above
RFC 2512: Accounting Information for ATM Networks	Same as above
RFC 2514: Definitions of Textual Conventions and OBJECT-IDENTITIES for ATM Management	Same as above

Table 2.2 Internet-Drafts edited by the IETF AToM MIB Working Group

Title	Date last updated
Definitions of Supplemental Managed Objects for ATM Management	March 13, 1998
Definitions of Textual Conventions and OBJECT-IDENTITIES for ATM Management	October 19, 1998
Definitions of Tests for ATM Management	September 1, 1998
Managed Objects for Controlling the Collection and Storage of Accounting Information for Connection-Oriented Networks	4 October 1998
Definitions of Managed Objects for ATM Management	October 19, 1998
Definitions of Managed Objects for the SONET/SDH Interface Type	October 19, 1998
Accounting Information for ATM Networks	4 October 1998
Managed Objects for Recording ATM Performance History Data Based on 15 Minute Intervals	25 November 1996

2.1.1 MIB-II (Interfaces for physical Layer, Cell Layer, AAL3/4, and AAL5)

It is defined in various MIB module definitions how to use the interface table to provide information about physical layer as well as higher layers (as logical interfaces), like ATM and AAL layers.

2.1.2 SONET/SDH MIB

SONET MIB [IETF1998g] [RFC1595] defines how to use the ifTable for SONET/SDH interfaces, including SONET interface itself (including section and line sublayer), which has 39 as its ifType value, SONET paths, which have 50 as their ifType value, and SONET VTs (Virtual Tributaries), which have 51 as their ifType value.

SONET MIB also defines current status and error statistics like Errored Seconds (ESs), Severely Errored Seconds (SEsSs), Severely Errored Framing Seconds (SEFSs), Coding Violations (CVs), Unavailable Seconds (UASs).

So this MIB module supports configuration management (ifTable entries), fault management (current status) and performance management (errored seconds, etc.).

2.1.3 ATM MIB (RFC2515)

The management functions supported by this MIB module [RFC2515] [IETF1998f] [RFC1695] are listed in the following table (The keyword *Information* under the column *Configuration* means the MIB object(s) provides configuration information, while the keyword *Control* means the MIB object(s) provides a way to configure the managed object through SNMP):

Table 2.3 Functions supported by the ATOM MIB

MIB Object(s)	Configuration	Performance	Fault	Accounting	Security
ATM Interface Configuration Parameter Group	Information/ Control				

Table 2.3 Functions supported by the AToM MIB

MIB Object(s)	Configuration	Performance	Fault	Accounting	Security
ATM Interface DS3 PLCP Group		SEFS/Alarm/ UASs	SEFS/Alarm/UASs		
ATM Interface TC Sublayer Group		OCD/Alarms	OCD/Alarms		
VPL Group	Information/ Control		Admin/Oper Status/Last change		
VPL Group	Information/ Control		Admin/Oper Status/Last change		
VP Cross Connect Group	Information/ Control		Admin/Oper Status(bi- direction)/Last change		
VC Cross Connect Group	Information/ Control		Admin/Oper(bi- direction) Status/ Last change		
AAL5 VC Connection Performance Stats Group		CRC Errors/ SAR time-outs/ Oversized SDUs	CRC Errors/SAR time-outs/ Oversized SDUs		

2.1.4 ATM Supplemental MIB

ATM Supplemental MIB [IETF1998a] defines additional objects to those defined in RFC 2515, e.g., signalling layer statistics, SVC tables. This draft also specifies how to represent a VPL (VP tunnel) as a logical interface and how to create it.

This MIB module also defines two traps for status change of a PVPC and PVCC.

The management functions supported by this MIB module are listed in the following table:

Table 2.4 Functions supported by the ATM Supplemental MIB

MIB Object(s)	Configuration	Performance	Fault	Accounting	Security
VPL SVC Cross-Connect Table	Information/ Control				
VCL SVC Cross-Connect Table	Information/ Control				
ATM Interface Signalling Stats Table		Yes	Yes	Yes	Yes
ATM Interface Signalling Support Table	Information/ Control				Yes
ATM Interface Administrative Address Table	Information/ Control				
VPI/VCI to Address Mapping Table	Information/ Control				
ATM Address to VPI/VCI Mapping Table	Information/ Control				
VPL Statistics Table		Cells In/Out, Discarded/ Tagged	Cells Discarded/ Tagged	Cells In/Out, Discarded/ Tagged	
VPL Logical Port Configuration Table	Information/ Control				
VCL Statistics Table		Cells In/Out, Discarded/ Tagged	Cells Discarded/ Tagged	Cells In/Out, Discarded/ Tagged	
AAL5 per VCC Stats Table		Total Errors, Pkts In/Out, Octets In/Out	Total Errors, Pkts In/Out, Octets In/Out	Total Errors, Pkts In/Out, Octets In/Out	
ATM Interface Configuration Extension Table	Information/ Control				

Table 2.4 Functions supported by the ATM Supplemental MIB

MIB Object(s)	Configuration	Performance	Fault	Accounting	Security
ILMI Service Registration Table	Information/Control				
ILMI Network Prefix Table	Information/Control				
Trap Control			Traps on PVPL/PVCL change		

2.1.5 Accounting MIB

There are two MIB modules for accounting: the accounting information MIB module [RFC2512] [IETF1998d] and the MIB module for controlling the collection and storage of accounting information [IETF1998c]. The first module defines MIB objects that can be collected for connection(s), while the second is used to control which objects are to be collected and how they will be logged to files.

The accounting information to be collected is not designed to be accessed by SNMP. Instead, it will be logged to files and these files can be accessed by other protocol (FTP, etc.). So it is not designed for real-time monitoring (not surprisingly).

This MIB module is designed for accounting management, although the data collected can be used for non-real time performance analysis as well.

2.1.6 Performance History MIB

Since the ATM Forum M4 MIB is publicly available and has equivalents for the most important objects in this specification, this draft [IETF1996] has been discontinued.

2.1.7 ATM Test MIB

ATM Test MIB [IETF1998b] defines several test types and ways to specify locations and read back test results. This can be used with the other MIB modules to do testing such as loopback test.

This MIB module is for fault management because the only tests are loopback tests, not performance tests.

2.2 The ATM Forum SNMP MIBs

2.2.1 ILMI MIB

The Integrated Local Management Interface [ATMF1996] (once called Interim Local Management Interface because it was expected that these procedures would be used for only a short period of time.) provides any ATM device (e.g., End-systems, Switches, etc.) with status and configuration information concerning the Virtual Path Connections, Virtual Channel Connections, registered ATM Network Prefixes, registered ATM Addresses, registered services, and capabilities available at its ATM Interfaces. The ILMI supports bi-directional exchange of ATM Interface parameters between two connected ATM Interface Management Entities (IMEs). There are four ILMI MIB modules: the Textual Conventions MIB, the Link Management MIB, the Address Registration MIB, and the Service Registry MIB.

2.2.1.1 Link Management MIB

The management functions supported by this MIB module are listed in the following table:

Table 2.5 Functions supported by the ILMI MIB

MIB Object(s)	Configuration	Performance	Fault	Accounting	Security
Physical Port Table	Information		Status		
ATM Layer table	Information				
ATM Layer statistics table		Cells received/ dropped/ transmitted	Cells dropped	Cells	
VPC Table	Information		Status		
VCC Table	Information		Status		

This MIB module also defines two traps for status change of VPC/VCC.

The ILMI MIB is designed mainly for communications between two ATM Interface Management Entities (IMEs) across a UNI interface, while ATM MIB is designed for providing information for an NMS (network management system) and contains more MIB objects. This is why the CNM, which is an approved specification of ATM Forum and will be described in details in Section 2.3 on page 19, references the ATM MIB instead of ILMI MIB.

2.2.1.2 ATM Address Registration MIB

This MIB module specifies MIB objects for IMEs on the network and user side to negotiate the ATM address, which is comprised of network prefix and end system address.

The management functions supported by this MIB module are listed in the following table:

Table 2.6 Functions supported by the ATM Address Registration MIB

MIB Object(s)	Configuration	Performance	Fault	Accounting	Security
Network Prefix Table	Information/Control				
ATM Address table	Information/Control				

2.2.2 ATM RMON MIB

Once a draft of IETF, the ATM RMON MIB is now an approved specification of ATM Forum.

ATM RMON [ATMF1997] can be used to configure an agent to monitor traffic/connections from an application/host (end system) /network and between applications/hosts/networks, seen at any group of ATM interfaces. Sources can be aggregated.

Currently only the number of calls and cells transferred (address-based) are collected. This requires that the interfaces be in the ifTable. While this allows ATM layer, AAL5, and VP tunnel (logical port) be monitored, it precludes VCCs be monitored. Actually the ATM RMON is not designed to collect per-connection accounting statistics, nor is it designed to collect aggregated statistics for a group of VCCs.

There is a set of counters for CBR and VBR, and another for ABR and UBR.

2.3 Customer Network Management (CNM) and M4 Management Interface

The Customer Network Management (CNM) and M4 management interfaces are two of the interfaces defined in the ATM Forum Network Management Sub-Working Group's ATM Network Management Reference Model, which is shown in Figure 2.1 on page 20. The model is for describing the various types of network management needed to support ATM devices, private networks, public networks and the interaction between them. In this model, M1 is the ATM management interface needed to manage an ATM terminal device. M2 is the management interface needed to manage a private ATM network. M4 is the management interface needed to manage a public network service, including both network element management and service management functions. M3 is the management interface needed to allow a customer to supervise their use of their portion of a public ATM network. M5 is the management interface needed for management interaction between two public network providers. M3 functions defined in [ATMF1994] include configuration, fault and performance management at this time. Private interfaces (M1 and M2) have largely been realized via the combination of ILMI, UNI, and RFC 2515.

- M1: Private Network Manager to End User
- M2: Private Network Manager to Private ATM Network
- M3: Private Network Manager to Public Network Manager (CNM, Customer Network Management interface)
- M4: Public Network Manager to Public ATM Network
- M5: Public Network Manager to Public Network Manager

Figure 2.1 ATM Forum's Management Model

While the M2, M3, M4 and M5 management definitions can provide a top down network wide view, they are not the only management functions relevant to ATM. The Integrated Local Management Interface (ILMI) provides an ATM link specific view of the configuration and fault parameters of a User Network Interface (UNI). The UNI also provides some layer management functionality via Operations Administration and Maintenance (OAM) cells. Since the ILMI allows information retrieval by customers from a public network, it can be considered a CNM service. However, the ILMI is embedded in the UNI and must be processed by

equipment directly connected to a UNI. In contrast, the M3 interface allows direct access by customer premises based management systems to the CNM service.

2.3.1 Approved specifications related to the model

There are 10 approved specifications related to the model shown in Figure 2.1, among them 2 are related to SNMP as shown in Table 2.7.

Table 2.7 Approved specifications related to CNM based on SNMP

Approved Specifications	Approved Date
Customer Network Management (CNM) for ATM Public Network Service (M3 Specification)	Oct, 1994
SNMP M4 Network Element View MIB	July, 1998

2.3.2 CNM M3 Specification

The M3 Specification [ATMF1994] defines the ATM management requirements and interface specifications for the M3 public network provider to private network manager. These M3 requirements are defined in two classes to allow public network providers to offer modular, incremental capabilities to meet different levels of customer's needs. Class 1, which is mandatory, provides monitoring information on the configuration, fault and performance management. While class 2, which is optional, provides control capability and subscription information.

2.3.3 SNMP M4 Network Element View MIB

This MIB module [ATMF1998b], which implements part of the logical MIB defined in another approved specification "*M4 Interface Requirements and Logical MIB: ATM Network*

Element View”, defines some experimental MIB objects that provide management system access to the ATM Forum’s M4 (Network Element View) interface via SNMP.

2.4 What Additional Information Would be Important

This additional information would be very helpful for detecting what the actual problem is:

- Traps on VP/VC Alarms (AIS). It will need control when trap will be sent (AIS/RDI, defect or failure).
- Test MIB for performance testing, using OAM.
- Protocol statistics for AAL1 and AAL3/4. These are already included in the M4 logical MIB, but not in SNMP MIB.
- Control traffic management (tagging vs. discarding on violations).

Chapter 3 Aspects of Management of ATM Networks

In this chapter, we will describe the five aspects of network management: configuration management, fault management, performance management, security management, and accounting management. Later we will show what we have done in the management of ATM LANs in Nortel.

3.1 Fault Management

3.1.1 Alarm surveillance and correlation

Alarm surveillance can be and should be done in multiple layers.

3.1.1.1 Physical layer

There are several alarms defined for physical layer management. Here are the important ones:

- **SONET/SDH**
 - **Loss of Signal (LOS)**
 - **Remote Defect Indication (RDI)**
 - **Alarm Indication Signal (AIS)**

- Loss of Frame (LOF)
- Loss of Pointer (LOP)
- Loss of Cell Delineation
- DS3 PLCP (Physical Layer Convergence Procedure/Protocol, IEEE 802.6)
 - Loss of Signal (LOS)
 - Loss of Frame (LOF)
 - PLCP LOF
- DS1/DS3 cell mapping
 - Loss of Signal (LOS)
 - Loss of Frame (LOF)
 - Loss of Cell Delineation

3.1.1.2 ATM layer

There are alarms defined for ATM (cell) layer management, the most important ones include:

- Alarm indication signal: VP-AIS and VC-AIS
- Remote defect indications: VP-RDI and VC-RDI

3.1.2 Fault localization and testing

This can be partially achieved by using OAM loopback cells, which are very useful for:

- Verify connectivity
- Isolate faults
- Perform pre-service acceptance tests

3.1.3 Alarm correlation

The alarms listed above should be correlated. This will be examined in details in Chapter 5.

3.2 Performance Management

3.2.1 Performance monitoring

Performance monitoring can be and should be done in multiple layers. Thresholds can be set to generate alarms when network performance is severely degraded.

3.2.1.1 Physical layer

Physical layers can be monitored for traffic statistics:

- DS3 PLCP frames transferred
- Cells (including cells for physical layer) transferred in DS1/DS3 cell mapping
- Cells (including cells for physical layer) transferred in SONET interfaces

3.2.1.2 ATM layer

Cells in/out can be monitored. VP/VC performance monitoring can be achieved by OAM performance test cells, which can be used to measure:

- Delay
- Cell delay variation
- Errored blocks
- Severely errored blocks
- Lost user information cells
- Mis-inserted user information cells
- Excessive cell transfer delay occurrences

3.2.1.3 Upper Layers

Upper layers like AAL layers, LAN emulation etc. can be monitored as well.

3.2.2 Protocol performance monitoring and test

Protocol can be monitored to see if the protocol is performing properly in a specific implementation. Thresholds can be set so that alarms will be sent when protocol performance is severely degraded.

3.2.2.1 Physical Layer

Overhead and frame processing can be monitored, here are the important ones:

- **SONET**
 - **BIP (Bit Interleaved Parity) Errors**
 - **FEBEs (Far End Block Errors)**
 - **FERF (Far End Receiver Failure)**
 - **Correctable HCS (Header Check Sequence) Error**
 - **Uncorrectable HCS Error**
 - **Errored Seconds (ESs)**
 - **Severely Errored seconds (SESSs)**
 - **Unavailable Seconds (UASs)**
- **DS3 PLCP**
 - **DS3 protocol performance**
 - **PLCP BIP (Bit Interleaved Parity) errors**
 - **PLCP FEBEs (Far End Block Errors)**
 - **PLCP RAIs (Remote Alarm Indications)**
 - **Cell HEC (Header Error Control) errors**

3.2.2.2 ATM layer

ATM cell header processing can be monitored, most importantly:

- Wrong VPI/VCI (out of range or not connected)
- Cells discarded due to header bit errors in other fields, this and the previous one are a good indication that the other side of the interface has problems with the ATM layer protocol.
- Cells dropped because of full buffer
- Connection attempts failed because of:
 - Resource shortage
 - QoS can not be met

Both of these two causes could be caused by wrong routing information, which in turn is caused by false information sent from other switches, memory shortage, and/or memory corruption.

- Connection set up delay

We can also monitor whether OAM cells are processed and transferred properly by monitoring:

- OAM cells discarded due to ATM layer header errors.
- OAM cells discarded due to header bit errors.
- OAM cells with wrong information in its information fields, this could be a security concern.

3.2.2.3 AAL Layer

Errors like invalid fields and incorrect fields are indications of protocol problems. So are:

- Invalid format of control (header/trailer) fields
- Oversized SDUs received
- CRC violations
- Reassembly failures (timeout, incomplete)

There are AAL type specific performance, most importantly:

- AAL1
 - Correctable SAR-PDU (Segmentation and Reassembly PDU) header errors
 - Uncorrectable SAR-PDU header errors
 - Bad sequence number (lost, mis-inserted cells)
 - Clock recovery failures (caused by lost cells)
 - AAL-1 SDT (Structured Data Transfer) Error (caused by lost cell, applications will be affected)
 - Cells arrives untimely (buffer overflow/underflow)
 - Cells discarded (because of SAR-PDU header error or bad sequence number)

- AAL3/4
 - Correctable SAR-PDU header errors
 - Uncorrectable SAR-PDU header errors
 - Bad sequence number
 - Reassembly failures (wrong ST/SN, LI, CRC)
 - Reassembly time-out
 - MID wrong (reassembly to wrong upper layer)
- AAL5
 - Length violations (lost cells)
 - CRC Error
 - Reassembly time-out

3.2.2.4 Upper Layers

Important alarms from upper layers include:

- (IP) Reassembly fails (checksum error or time-out)
- (TCP) Retransmissions
- Unavailable (TCP/UDP) port or SAP (service access point)
- Wrong destination (for other IP or non-existing IP)
- Application glitches
- Application performance degradation

3.2.3 Network traffic monitoring

Capabilities needed include:

- Support needed for all congestionable modules
- Measures of congestions
- Thresholding capabilities
- Network traffic management surveillance measurements
- Network traffic management controls

3.3 Configuration Management

Capabilities needed include:

- System configuration for switches (memory update/query/backup/restoration, software update/configuration)
- Setting up and tearing down PVCs (VPC and VCCs)
- Changing traffic descriptors
- Visualizing VP/VC connections

The SNMP protocol (version 1) is not suitable for this because of its in-secure nature.

3.4 Accounting Management

Capabilities needed include:

- Traffic statistics for connections, and traffic statistics from/to a server
- UPC (Usage Parameter Control) conformance
- Actual QoS

3.5 Security Management

Capabilities needed include:

- Traffic control to protect network from non-conforming sources
- Preventing break-ins

3.6 Practical experience in managing ATM LANs

3.6.1 Work done related to ATM network management

A network management toolset, MAGNet, was used to manage three ATM networks in Nortel Networks. The main purpose is to generate reports for network engineering staff.

Alarm correlation was not done for this network, largely because that the alarms mentioned before are either not reported or not accessible from the ATM switches in these networks.

This section explains what has been done for management of these networks.

3.6.2 The topology of the networks

The topology of the networks is shown in Figure 3.1. Although the networks differ in number of switches, they share some commonalities:

- One of the switches has uplink to a Cisco router, which is connected to the campus backbone (FDDI).
- There are some servers (HP and/or Sun) directly connected to switch(es).
- There are Fore ES3810 connected to switches. On a Fore ES3810 there are some 10/100M Ethernet ports, which provide dedicated Ethernet.

All switches are from Fore Systems, including Fore ASX 200 and ASX1000.

Other network devices include Fore's ES3810, Cisco routers, Sun servers, and HP servers, all of them have ATM interface(s). All devices are responding to SNMP queries, thus all management work done is based on SNMP.

Fore ATM switches have good support of ATM Forum's ILMI MIB, but not MIB II interfaces group.

Security management and accounting management were not done. The main reason is that the ATM networks are LANs and security and accounting are less as an issue. The SNMP agents are not supporting these two functionalities, either.

Figure 3.1 Topology of the networks

3.6.3 Configuration Management

The work related to configuration management includes:

- configuring NM (network management) console and data collectors so the ATM LAN configuration will be presented in GUI.
- writing scripts to generate *ad hoc* configuration report.

The configuration of ATM devices, like setting up VPC/VCC, is not done. The main reason is we were not authorized to do such tasks, and the SNMP agents in the ATM LANs do not support this functionality (configuration of switches through SNMP).

3.6.3.1 NM Console configuration management

A script is written to discover and configure the NM console to visualize the ATM LANs automatically so labor-intensive NM console configuration can be avoided, this include:

- **Network element configuration:** The MIB object `sysObjectID` is retrieved. Then certain level of components (ATM ports, VPCs, and VCCs) are configured.
- **Network topology and element auto-discovery:** The network topology auto-discovery is enabled by the following SNMP MIB objects:
 - *portILMIRemoteIpAddress* (Fore's private MIB object) and *atmfPortMyIfName* reported by Fore's ATM switches. They tell what the IP address of the another end of an ATM links is.
 - *ifType* reported by workstations and Cisco routers, which tells which interface is an ATM interface.
 - *ipRouteTable* reported by Cisco's routers which have an excellent support of this table.

This table tells which IP is directly connected to an interface of a Cisco router.

Notice that when there are multiple ATM interface on workstations, or there are more than one ATM link between two ATM switches, the topology discovery gets more complicated, i.e., it has to be resolved that which interface on one switch is connected

to which interface on another ATM switch. Actually this is done by manual intervention.

- **VPC/VCC visualization:** Unfortunately, it is impossible to configure the network model automatically to show VPC/VCC, because no agent is reporting the VP or VC cross connect table. So only VPL/VCLs are shown in xnetmon, which is the graphic network management console of MAGNet.

3.6.3.2 Real time configuration reporting

A script was written to report some configuration information which is more dynamic, such as virtual path/virtual channel connections. A sample output of the script is shown in Figure 3.2.

IP address: 147.4.85.69 vendor: Fore Systems

VPC Table (Port 3)						
VPI #	OperStatus	TxTrafficType	RxTrafficType	QOS Category	TxQoS Class	RxQoS Class
0	end2endUp	NoDescriptor	NoDescriptor	determinist	0	4

VCC Table (Port 3 VP 0)						
VCI #	OperStatus	TxTrafficType	RxTrafficType	QOS Category	TxQoS Class	RxQoS Class
3	localUpEnd2e	NoDescriptor	NoDescriptor		0	4
4	localUpEnd2e	NoDescriptor	NoDescriptor		0	4
5	localUpEnd2e	NoDescriptor	NoDescriptor		0	4
14	localUpEnd2e	NoDescriptor	NoDescriptor		0	4
15	localUpEnd2e	NoDescriptor	NoDescriptor		0	4
16	localUpEnd2e	NoDescriptor	NoDescriptor		0	4

Figure 3.2 A sample ad hoc configuration report

3.6.3.3 Non-real time configuration reporting, graphing

Another script is written to generate an NPMS (Network Performance Management System) reports which tell how many VPC/VCC were configured and their peak, prime time, prime

time peak value. The MIB objects *atmfAtmLayerConfiguredVPCs* and *atmfAtmLayerConfiguredVCCs* are collected for this purpose. A sample report is shown in Figure 3.3.

MAGNet 5.5 NPMS REPORT Daily ATM Metrics Report for Classes: AtmLayer.ForeASX1000 Atm-
Layer.ForeASX1000_WAN AtmLayer.F

Produced: Tue Dec 29 11:41:32 1

Object	Link Connected to	VPCs				VCCs			
		Peak	Prime	Prime	Pk	Peak	Prime	Prime	Pk
1A1-atmLayer.tcaratsd.c -		1	1	1	1	6	6	6	6
1A2-atmLayer.tcaratsd.c -		1	1	1	1	14	16	14	15
1A3-atmLayer.tcaratsd.c -		1	1	1	1	14	17	14	16
1A4-atmLayer.tcaratsd.c -		1	1	1	1	15	16	15	16
1B1-atmLayer.tcaratsd.c -		1	1	1	1	21	22	21	22
1B2-atmLayer.tcaratsd.c -		1	1	1	1	39	39	39	39
1B3-atmLayer.tcaratsd.c -		1	1	1	1	21	21	21	21
1B4-atmLayer.tcaratsd.c -		1	1	1	1	15	16	15	16
(snip)									
1D3-atmLayer.rgtyafsw.g -		1	1	1	1	6	6	6	6
1D4-atmLayer.rgtyafsw.g -		1	1	1	1	14	14	13	14
1E2-atmLayer.rgtyafsw.g -		1	1	1	1	11	13	10	12
1E3-atmLayer.rgtyafsw.g -		1	1	1	1	39	40	39	40
1E4-atmLayer.rgtyafsw.g -		1	1	1	1	5	5	5	5
2A1-atmLayer.rgtyasw2.g -		1	1	1	1	14	15	14	14
(snip)									
4D1-atmLayer.rgtyasw4.g -		1	1	1	1	6	6	6	6
4D2-atmLayer.rgtyasw4.g -		1	1	1	1	6	6	6	6
4D3-atmLayer.rgtyasw4.g -		1	1	1	1	6	6	6	6
4D4-atmLayer.rgtyasw4.g -		1	1	1	1	6	6	6	6
4E1-atmLayer.rgtyasw4.g -		1	1	1	1	5	5	5	5
4E2-atmLayer.rgtyasw4.g -		1	1	1	1	4	4	4	4
4E3-atmLayer.rgtyasw4.g -		1	1	1	1	8	9	8	9
Total		110	110	110	110	1325	1366	1322	1354

DEFINITIONS FOR THE ABOVE FIELDS

Connected To	Name of connected object.
Object Name	Fully qualified object name.
VCCs	Number of configured Permanent VCCs seen in the end of the day.
VCCs Peak	Peak number of configured Permanent VCCs seen in the end of the day.
VCCs Prime	Number of configured Permanent VCCs seen in the prime time of the day.
VCCs Prime Pk	Peak number of configured Permanent VCCs seen in the prime time of the day
VPCs	Number of configured Permanent VPCs seen in the end of the day.
VPCs Peak	Peak number of configured Permanent VPCs seen in the end of the day.
VPCs Prime	Number of configured Permanent VPCs seen in the prime time of the day.
VPCs Prime Pk	Peak number of configured Permanent VPCs seen in the prime time of the day

Figure 3.3 Sample daily configuration report

3.6.4 Performance Management

3.6.4.1 MIB objects related to performance management

Fore's SNMP agent does not reports ifEntry for ATM interfaces. But it does report some ATM Forum's and Fore's proprietary MIB objects which can be used for performance management. E.g.,

- *portMaxBandwidthIn* and *portReceivedCells*. They are much like ifInOctets/ifSpeed. The input utilization can be calculated by the following formula:

$$\text{utilization} = \text{delta}(\text{portReceivedCells}) / (\text{portMaxBandwidthIn} * \text{delta}(\text{sysUpTime}))$$
- *portMaxBandwidthOut* and *portTransmittedCells*, this is much like ifOutOctets/ifSpeed, the output utilization can be calculated by the following formula:

$$\text{utilization} = \text{delta}(\text{portTransmittedCells}) / (\text{portMaxBandwidthOut} * \text{delta}(\text{sysUpTime}))$$
- *portAllocBandwidthIn* and *portAllocBandwidthOut* reports how much bandwidth has been allocated (to VPC/VCC). These tell how much bandwidth has been allocated (subscribed). These will also tell how the sources are compliant with the traffic contracts when compared with *portReceivedCells* and *portTransmittedCells*.
- *atmfAtmStatsReceivedCells*, *atmfAtmStatsDroppedReceivedCells*, *atmfAtmStatsTransmittedCells*. These tell how large the traffic volume is. These also give the dropped cell ratio:

```
CellLostRatio = delta(atmfAtmStatsDroppedReceivedCells) /
delta(atmfAtmStatsReceivedCells)
```

This does not tell what the cell lost ratio of a connection is, but it gives a rough idea of how network is behaving. Also notice that these three MIB objects have been obsoleted in version 4 of ATM Forum UNI specification.

- There are also some MIB objects related to physical layer performance, the important ones include (They are all Fore's proprietary MIB objects):
 - sonetSectionBIPs
 - sonetSectionLOSs
 - sonetSectionLOF
 - sonetLineBIPs
 - sonetLineFEBEs
 - sonetLineAISs
 - sonetLineFERFs
 - sonetPathBIPs
 - sonetPathFEBEs
 - sonetPathLOPs
 - sonetPathAISs
 - hwPortOverflows

Workstations, when Fore's agent (rather than Sun or HP's) is enabled, are also reporting MIB-II plus ATM Forum's ILMI MIB.

Cisco's router are reporting standard MIB-II, so status and utilization information can be got from the interface group in MIB II.

3.6.4.2 Event and alarm generation (performance thresholding)

Some thresholds were set for MIB objects which report physical layer performance.

3.6.4.3 Performance reporting and graphing

A script called mperl was written to:

- Read a Perl-like script which specifies MIB objects to be retrieved and what should be done to process the MIB objects.
- Certain MIB (objects) specified by the script will be retrieved at a customizable interval.
- Generate performance report or graph as specified by command options specified for the script.

A sample mperl script is shown Figure 3.4.

```

$incells = ${"delta_portReceivedCells.$ifID"};
$bandwidth = ${"portMaxBandwidthIn.$ifID"};
$secs = $delta_sysUpTime_0 / 100;
$now=`date`;
printf "$now";
if ($bandwidth && $secs) {
printf "The in utilization of $ARGV[0] is %2.2f\n", (100 * $incells /
$bandwidth / $secs);
}
$outcells = ${"delta_portTransmittedCells.$ifID"};
$bandwidth = ${"portMaxBandwidthOut.$ifID"};
$secs = $delta_sysUpTime_0 / 100;
if ($bandwidth && $secs) {
printf "The out utilization of $ARGV[0] is %2.2f\n", (100 * $outcells /
$bandwidth / $secs);
}
printf "\n";

```

Figure 3.4 Sample mperl script

The output of the script is shown in Figure 3.5.

```

Tuesday December 29 13:49:31 EST 1998
The in utilization of Port-1C1 is 0.18
The out utilization of Port-1C1 is 0.15

Tuesday December 29 13:49:41 EST 1998
The in utilization of Port-1C1 is 0.18
The out utilization of Port-1C1 is 0.10

Tuesday December 29 13:49:51 EST 1998
The in utilization of Port-1C1 is 0.21
The out utilization of Port-1C1 is 0.10

Tuesday December 29 13:50:01 EST 1998
The in utilization of Port-1C1 is 0.19
The out utilization of Port-1C1 is 0.10

Tuesday December 29 13:50:11 EST 1998
The in utilization of Port-1C1 is 0.27
The out utilization of Port-1C1 is 0.32

Tuesday December 29 13:50:21 EST 1998
The in utilization of Port-1C1 is 0.21
The out utilization of Port-1C1 is 0.32

```

Figure 3.5 Sample output from mperl script

Following tools (based on mperl) have been configured so they can be invoked from xnetmon:

- cells received (graph)
- cells transmitted (graph)
- In utilization (graph)
- Out utilization (graph)
- In utilization (report)
- Out utilization (report)

3.6.4.4 Performance data collection/reduction and reporting/graphing

All above mentioned MIB objects are also configured to be collected by NPMS.

3.6.5 Fault Management

Following MIB objects are used to monitor status of ATM ports and VPL/VCLs:

- *portOperStatus* (Fore's proprietary MIB object) for status of an ATM port.
- *atmfVpcOperStatus* for status of a VPL.
- *atmfVccOperStatus* for status of a VCL.

Chapter 4 Alarm Correlation Algorithms

4.1 Introduction

From now on, we concentrate on alarm correlation in ATM networks. First in this chapter, we discuss two algorithms for alarm correlation: Katzela-Schwartz's [Katzela1995] and Brunet's [Brunet1998].

4.2 Katzela-Schwartz approach

4.2.1 Definition of the problem and model

The problem the two authors try to solve can be formulated as follows:

Given a model for the communication system and a set of observed alarms how does one design an algorithm that can create a number of fault hypotheses and assign a measure of confidence in each of them according to a belief metric?

The requirements are: The algorithm should be simple, should have low order polynomial computational complexity, should provide quick and accurate response, and should be easily adapted to a distributed environment. It should handle effectively different fault scenarios and it should be robust to incomplete, redundant, unreliable and contradictory information.

In Katzela-Schwartz's approach [Katzela1995] [Katzela1996], networks are modeled as follows. A telecommunication network consists of a number of managed objects, referred to from now on simply as objects or entities. An object is a "part" of the network that has a separate and distinct existence. An object can be a network, a node, a layer in a protocol stack, a software process, a virtual link, a physical element like an optical fiber, a hardware component, etc. Objects in a communication system consist of other objects down to the level of the smallest objects which are considered indivisible. An indivisible object is defined as a *terminal object*. A *terminal object* is defined as *active* if it is in use during the time the system is observed.

Objects in a telecommunication network are dependent upon each other in rather complex ways. Most of the time a failure in one object has side effects on other objects that are dependent on it. For example, a link failure has an effect on other resources in the network, e.g., connections on the various layers which use the link will experience time-outs. The knowledge of these dependencies is very important for the fault diagnosis process since it provides information for the way that faults and their effect propagate in the network.

The proposed system model is a directed graph $G = (E, D)$ called the *dependency graph*, where E is a finite, non-empty set of *active terminal objects* e_i , and D a collection of ordered pairs of distinct objects from E . A directed edge $(e_i, e_j) \in D$ denotes the fact that a fault in e_i has a side effect on e_j , i.e., e_j is dependent on e_i .

Each vertex e_i is assigned a weight p_i , which is the probability that the object e_i fails independently of the state, failed or not failed, of any other *active terminal object* that is

dependent on it. Each directed edge (e_j, e_i) is assigned a weight p_{ji} which represents the strength of dependency between the entities it connects. Specifically, p_{ji} is the conditional probability $p_{ji} = P(e_j \text{ fails} \mid e_i \text{ fails})$ that entity e_j fails as a result of the failure of entity e_i . In addition, the weight of a directed path is defined as the product of all the weights p_{ij} of the edges that compose the path. In other words, if $\hat{P} = v_1 v_2 \dots v_n$ is a directed path in the dependency graph, the weight of $w(\hat{P})$ is defined as the product of the weights of the directed edges that form the path, basically $w(\hat{P}) = \prod_{i=1}^n p_{i(i+1)}$. Finally, the strength of dependency p_{kl} between two non-adjacent vertices e_k and e_l , is defined as the maximum of the weights of all directed paths from e_k to e_l . In practice the probabilities p_i and p_{ji} for all i, j can be estimated from system specification information, history of previous failures, etc.

As an example consider Figure 4.1(a) which represents the topology of a network that consists of three nodes and four links. Each of these objects is considered, for the selected level of abstraction, a *terminal object*. Assume that during our observation there are active (established) connections from node A to node B via the link L_1 , from node A to node C via link L_3 , and from node C to node B via link L_2 , while link L_4 is not in use. The corresponding dependency graph is depicted in Figure 4.1(b) together with the correspondence between the real objects and the vertices in the graph.

Figure 4.1 (a) Network Example (b) Dependency Graph

The system representation above has certain advantages. It provides *generality* because different types of communication systems can be easily modeled. It is *flexible* in the sense that the representation can be easily modified as the active picture of the observed system changes. It is *simple* because it gives a manageable model but at the same time it has the necessary complexity to represent real systems. Finally, it has the property of *similarity* which means that similar systems have similar representations. If, for example, another entity is added in the graph the representation changes only slightly. All the previous properties make the dependency graph a good system representation for the purposes of alarm correlation.

But as Brunet [Brunet1998] pointed out, it has disadvantages too: for example, it presupposes the knowledge of the probability of failure of a terminal object.

Faults manifest themselves as alarms and alerts which are emitted by system objects that are affected by the fault. It is assumed that an alarm carries only the identity of the object that sensed the malfunction. This is the minimum possible information an alarm could carry and therefore it is always present in any type of alarm from any real system.

Each alarm a_i , emitted by e_i , is characterized by the domain of the alarm $D(a_i)$, which is defined as the set of objects that might have caused the alarm, i.e., the set of objects which might be the primary or secondary source of the observed failure. The primary source of failure is defined as an object which is in fault and secondary source of failure is defined as an object which is malfunctioning (but not in fault) due to its dependency on another object which is a primary fault.

In addition a cluster of alarms is defined as the set of alarms whose domains intersect with each other. An alarm belongs to a cluster if its domain has an intersection with a domain of at least one alarm that belongs to this cluster. Two different clusters share no intersection. An example is given in Figure 4.2. The six alarms, as shown, were received. Each is associated with a number of objects. Four of the alarms share an intersection and thus belong to the same cluster (cluster 1). The remaining two also share an intersection and create another cluster (cluster 2).

Figure 4.2 Alarm Clusters

The obvious question is how to find the domain of an alarm. It was proposed to associate an alarm with all the objects that are related to the one that generated the malfunctioning indication. One can claim that almost all objects in a network are related in some fashion so it is pointless to attempt considering all such relations. Katzela adopts a model that associates an alarm a_i both with all the active objects that are most probably dependent on e_i and the active objects on which e_i is most probably dependent. The *dependency graph* representation will help to find these most probable objects. Suppose that the object e_i emits an alarm. This alarm will be associated with all the objects e_j in the dependency graph, which have dependency weights $p_{ij} \geq W$, where W is a parameter.

By choosing different values of W one can create different associations of entities with a given alarm. If W is too large only a few entities will be associated with the observed alarm. If

W is too small then almost all the entities in the dependency graph will be associated with the observed alarm. The selection of W is system and application dependent. As an example, suppose one chooses W as the minimum of all the edge weights in the dependency graph. Such a model is called the single linkage or nearest neighbor model, where an alarm a_i is associated with all entities e_j connected directly to e_i (the nearest neighbors) plus a few others which are within dependency strength W from e_i .

The problem of finding the domain $D(a_i)$ is a variation of the single source problem. The original single source problem finds the minimum cost paths from a source vertex in a graph to all other vertices in the graph. There are known algorithms which solve the problem in polynomial time, Dijkstra's algorithm, for example, with complexity $O(|E| \cdot |D|)$. In this case the cost of a path is the product of the weights of all the edges that compose the path and the problem is to find the maximum cost paths which are also greater than W . It is easy to transform the problem of finding the domain of an alarm to a variation of a single source problem if instead of the weight p_{ij} the equivalent $-\log(p_{ij})$ is used as cost for the path from e_i to e_j , subject to the restriction that only the minimum cost paths which have $\text{cost} \leq -\log(W)$ are included.

4.2.2 Fault identification

Assume that at a given time a number of alarms appear in the network. The objective is to design algorithms that are able to propose accurate hypotheses about the fault(s) that produced the observed alarms. By using the single linkage model the domain of each of the observed alarms can be computed.

Any object that appears in the domain of an alarm is a possible explanation of it. If two or more alarms belong to the same cluster, these alarms should be examined together because it is more likely that they are explained by the same set of faults. If the received alarms create more than one cluster then for each cluster we find the set of faults that best explain it. The union of all such sets, one for each cluster, gives the best explanation of the received alarms. In the remainder of the section it is assumed, for simplicity, that all the observed alarms belong to the same cluster.

A cluster of alarms A is associated with a set S , $S = \bigcup_{\forall a_i \in A} D(a_i)$, of N objects. For each object e_i in S the probabilities p_i and p_{ij} , as defined in the *dependency graph* representation, $\forall j$ such that $e_j \in S$, are assumed known. In other words, a cluster of alarms together with the set S is nothing more than the transitive closure of a connected sub-graph G_i of the dependency graph G which represents the observed system. The G_i can be found from the dependency graph by the single linkage model presented in Section 4.2.1.

Each cluster of alarms A may have a number of explanations. The objective of any fault localization algorithm is to find the best explanation among them. In this interpretation the best explanation of the received alarms is the minimum subset of S which has the following two properties:

- The entities in the chosen subset explain all the alarms in the cluster and
- maximize the probability that at least one of the objects in it is in fault.

For a set $S' \subseteq S$ of $k < N$ objects the probability that at least one entity in S is in fault is defined as:

•Equation 1

$$\begin{aligned}
 P(S') &= P(\text{at least one fault in } S') = 1 - P(\text{no faults in } S') \\
 &= 1 - \prod_{\forall i, (e_i \in S')} [1 - P(e_i \text{ a primary fault}) - P(e_i \text{ a secondary fault})] \\
 &= 1 - \prod_{\forall i, (e_i \in S')} \left[1 - p_i - \left(\sum_{\forall j, (e_j \in S, (j \neq i))} p_{ij} \cdot p_j + \epsilon \right) \right]
 \end{aligned}$$

The ϵ corresponds to the probability of all the events that lead to a secondary failure due to the concurrent primary failure of two or more entities. The p_{ij} 's and p_i 's are considered very small numbers so with small error $P(S')$ can be approximated as follows:

•Equation 2

$$P(S') = \sum_{\forall i, (e_i \in S)} \left(p_i + \sum_{\forall j, (e_j \in S, (j \neq i))} p_{ij} \cdot p_j \right)$$

It was proved by Katzela that the problem of finding the best explanation of the received alarms is NP-complete. The subsequent sections illustrate three methods for finding the best explanation.

4.2.3 Exhaustive search

The easiest straight-forward approach to find the best explanation of the received alarms would be the exhaustive search solution in which one:

- First forms all the possible subsets of the set S
- Then selects the minimum subset S_i of S that explains all the received alarms. If there is more than one such subset, selects the one which exhibits the maximum probability that at least one object in the subset is in fault.

This algorithm always guarantees to output the best explanation of the observed alarm cluster but since it examines all the 2^N possible subsets of S , it has exponential complexity $O(2^N)$, which makes the approach undesirable, except for small size sets.

4.2.4 Greedy search

The exhaustive search technique can be improved in the following way: The bulk of the sets formed in the first step of the exhaustive search algorithm are not even explanations of the observed alarms. Suppose that instead of examining all the subsets of S one could find a way to examine only a portion of them and still be able to find the best explanation of the received alarm cluster. This leads to the following algorithm:

- Start with the set S of objects associated with the alarm cluster.
- Find all the partitions of the set which consist of two disjoint sets of equal size.

- Of the resulting subsets select the one which explains all the alarms and has the maximum probability that at least one of the objects in the selected set is a fault.
- Apply the previous two steps recursively to the selected set until the resulting sets of the current partitioning are either singletons or neither of them could explain all the alarms.

It is easy to show that this algorithm will output the best explanation of the received alarms.

If in step k of the algorithm the set of entities being selected for partitioning contains K entities then $\binom{K}{K/2} \leq O\left(2^{(N/2)\log\left(\frac{4N}{N+2}\right)}\right)$ different possible partitionings can be formed. The maximum number of steps for the algorithm would be $\log N$ and the complexity $O\left(\log N 2^{(N/2)\log\left(\frac{4N}{N+2}\right)}\right)$. The reduction of complexity observed in this second algorithm is due to the idea of partitioning the set in each step of the search. Although the complexity is lower than the exhaustive search, it remains exponential in the number of objects in S . This is because in the second step of the algorithm all the possible partitions of the set are formed in order to guarantee that the algorithm always finds the best explanation of the alarms for the current stage of search.

4.2.5 Divide and Conquer

The problem of finding the best explanation of an observed alarm cluster is, as proved in Katzela's thesis, NP-complete. In general there are two approaches to get around the NP-completeness of the problem. First if the size of the set S is small the exhaustive search approach or the greedy one may be perfectly satisfactory. Second, for large inputs it may be possible to find a near optimum solution in polynomial time with the use of a heuristic algorithm. So an algorithm, called the *Divide and Conquer Maximum Mutual Dependency*

algorithm, was proposed. The main idea behind the algorithm is to partition the initial problem into smaller parts, find solutions for the parts, and then combine the solutions for the parts into a solution for the whole. The algorithm is summarized below, with more formal details appearing afterwards.

- Start with the set S of objects associated with the alarm cluster.
- Find the partitioning of the set S which results in two disjoint sets so that in each set the objects exhibit the maximum mutual dependency, i.e., for every object e_i and e_j that belongs to the same set and e_k that does not belong to the set, the dependency weight p_{ij} is greater than the dependency weights p_{ik} , p_{ki} , p_{jk} and p_{kj} .
- Of the two sets select the one which explains all the received alarms and has the maximum probability that at least one of the entities in the set is a fault. If there is no such set then find the subset of alarms that each set explains and select both.
- Apply the previous two steps of the algorithm recursively to the selected set(s) until the resulting sets of the current partitioning are singletons.

The main goal of the algorithm is to narrow the set S to its smallest possible subset which explains all the received alarms and has the maximum probability that at least one object in the set is in fault. The action in each stage of the algorithm aims at eliminating some of the elements in the set S as unlikely candidates of being faults. By partitioning with the maximum mutual dependency criterion the algorithm groups together objects which are highly dependent on each other. This means that it is more probable that only a small portion of the objects in each of the resulting sets represents primary faults. If both of the sets resulting from the partitioning

explain all the alarms then the algorithm selects the one which has the maximum probability that at least one of the elements in the set is in fault. This seems to be the most promising set, according to the criterion of optimality, to continue in the next stage. It has also been shown in Katzela's thesis, when the probabilities of independent failure for all the objects associated with the received alarm cluster are the same, the selected set has the maximum probability among all the sets of the same cardinality. As a consequence, in that case, there is even a stronger indication that the selected set is the most promising to continue the search in the next stage of the algorithm. This indication was also verified from the simulation results.

In the case that neither of the sets explains all the alarms, there is a good indication that there might be multiple faulty objects, some included in one set and some in the other. That is why the algorithm continues the search, in the next stage, in both sets (taking into account for each of them the alarms it explains). The above algorithm always gives an explanation of all the alarms in the observed alarm cluster. But it is easy to observe it does not always guarantee to give the best explanation of them. The reason is that it is possible, in some stage of the algorithm, that the maximum dependency partition results in two sets neither of which is the best explanation of the received alarms for that stage. Here are the details of the algorithm:

4.2.5.1 Input

- A cluster A of received alarms during the time interval T .
- A set S of N entities associated with the received alarms. The association is produced by the single linkage alarm association model presented in Section 4.2.1.
- For each object e_i in S the probabilities p_i and $p_{ij} \forall j$ such that $i \neq j$ and $e_j \in S$ are given.

4.2.5.2 Output

The smallest subset of S which explains the alarm cluster and has the highest probability that at least one of the entities in the set is a fault.

4.2.5.3 Method

The algorithm runs in two phases. *Phase I* is as follows:

- 1 At the first step each entity in the set S is treated as a group with one member. Hence, in the first step of the algorithm there are $m = N$ one-member sets, where N is the number of entities in the set S .
- 2 Examine all the pairs of the m groups and find those two which are mutually closest together (the entities in the sets have the maximum mutual dependency). These groups are merged together to form a new group in a higher level in the hierarchy. Calculate the probability that at least one entity in the new set is a primary fault. Calculate the dependency of the new group to all other groups.
- 3 Repeat Step 2 for $m = N$

Phase II: In the second phase the algorithm finds the smallest subset S_{sol} of the set S which explains all the alarms and has the maximum probability that a primary fault is in it.

- *Initialization:* Initially the solution set S_{sol} is empty. Apply a typical step of the algorithm with input pair (S, A) , where S is the set of entities associated with the received alarm cluster, A the set of alarms in the alarm cluster.
- *Typical Step:* The input in a typical step of the algorithm is the pair (S^i, A^i) where S^i is a subset of the set S and A^i is the subset of alarms from the received alarm cluster A that the

set S^t explains. From Phase I the two children of S^t , S_l^t and S_r^t , together with their probabilities $P(S_l^t)$, $P(S_r^t)$ are known. The action in the typical step is as follows:

- If the set S^t is a singleton add the entity of S^t to the solution set S_{sol} and stop.
- If S^t is not a singleton, and both its left and right children explain all the alarms in A^t , select the one which exhibits the higher probability that at least one entity in the set is a primary fault. The typical step will be repeated for the selected set.
- If S^t is not a singleton and one of its children explains all the alarms, and exhibits the higher probability, among the two children, that at least one entity in the set is a primary fault, then repeat the typical step for that child.
- If S^t is not a singleton, and only one of its children explains all the alarms but the other has the highest probability that at least one entity in the set is a primary fault, or neither of its children explain all the alarms, then:
 - Find the child which exhibits the higher probability, among the two children, that at least one entity in the set is a primary fault.
 - Find the subsets $A^{t'}$ of alarms from A^t that the child explains. The typical step is repeated for the child with input alarm set $A^{t'}$.
 - Repeat the typical step for the other child with input alarm set $A^t - A^{t'}$.

4.2.5.4 Example

We now include a simple example from Katzela's thesis in order to clarify the proposed algorithm. Suppose that we are given a simple communication network the dependency graph

of which is given in Figure 4.3(a). Let us assume that object e_1 failed and as a result alarms from the objects e_2 , e_3 and e_4 were observed. The corresponding alarm cluster is computed by the single linkage model with $W = 0.01$ and it is given in Figure 4.3(b).

Figure 4.3 (a) Dependency Graph (b) Alarm Clusters

The steps of phase I of the divide and conquer algorithm are given in Figure 4.4. The probabilities of the different sets are computed with the use of Equation 2 on page 51. The steps of phase II are given in Figure 4.5. For the second phase, initially the solution set S_{sol} is empty. The input set for the first step of the selection phase is the pair $S^t = (e_1, e_2, e_3, e_4, e_5, e_6)$, $A^t = (A_2, A_3, A_4)$. From the hierarchical representation in Figure 4.4 the left and right children of S are respectively: $S_l^t = (e_1, e_2, e_3, e_4)$ and $S_r^t = (e_5, e_6)$. Since $P(S_l^t)$ is greater than $P(S_r^t)$ and S_l^t explains all the received alarms, the set S_l^t is selected to continue the search in the next stage. The input for step two is the pair $S^t = (e_1, e_2, e_3, e_4)$, $A^t = (A_2, A_3, A_4)$. The left and right children of the set are respectively: $S_l^t = (e_1, e_2)$ and $S_r^t = (e_3, e_4)$. Both sets explain all the received

alarms but $P(S_l^t)$ is greater than $P(S_r^t)$. Thus the set S_l^t is selected to continue the search in the next stage of the algorithm. The input for step 3 of the algorithm is the pair $S^t = (e_1, e_2)$, $A^t = (A_2, A_3, A_4)$. The left and right children of the set are respectively: $S_l^t = (e_1)$ and $S_r^t = (e_2)$. Since $P(S_l^t)$ is greater than $P(S_r^t)$ and S_l^t explains all the received alarms the set S_l^t is selected to continue the search in the next stage. Finally the input for step 4 of the algorithm is the pair $S^t = (e_1)$, $A^t = (A_2, A_3, A_4)$. Since $S^t = (e_1)$ is a singleton, e_1 is added in the solution set S_{sol} and the algorithm stops. The algorithm suggests as the most probable object that fails, given the received information, the object e_1 (which in that case is the one which actually fails).

4.2.5.5 Performance Analysis

It is easy to show that the complexity of Phase I is polynomial $O(N^3)$ in the number N of entities in the set S . The complexity of Phase II has a lower bound of $O(\log N)$ and an upper bound $O(N)$. Thus the worst time complexity of the mutual dependency algorithm is no more than $O(N^3)$. Katzela has also shown the relative error of the problem is bounded by:

- Equation 3

$$\left| \frac{P(S_{sol})}{P(S^*)} - 1 \right| \leq \left| \frac{P(S_{sol})}{E[n^*]^2 P_{min}} - 1 \right|$$

Where S_{sol} is the solution that the algorithm gives and S^* the optimum solution from the exhaustive or greedy search, P_{min} is the minimum of all the probabilities, both dependent and independent, of the objects in S , and $E[n^*]$ is the expectancy of the number of objects in S^* .

Figure 4.4 Phase I of the Maximum Mutual Dependency Algorithm

Figure 4.5 Phase II of the Maximum Mutual Dependency Algorithm

4.3 Brunet's algorithm (Temporal model)

Some description of Brunet's algorithm is included here for reference in later chapters.

More details can be found in Brunet's thesis [Brunet1998].

4.3.1 Definition of the problem and model

Brunet's algorithm is designed to fix several problems of Katzela's algorithm caused by the fact that the latter is solely probability-based:

- The accuracy of the diagnosis relies exclusively on how accurate the probabilities of faults and causal relations are. This would be fine if the probabilities could be estimated. However, in practice, this is most likely not true: it is very hard to estimate the probabilities. Thus the algorithm will lead to wrong diagnosis.
- The time when the alarms are sent are not factored in diagnosis. Thus it would be impossible to find out the right cause if there are cycles in causal model. E.g., if a fault in *A* will lead to a fault in *B*, a fault in *B* in turn will lead to a fault in *C*, and a fault in *C* will cause a fault in *A*, and the probabilities of fault happen independently in *A*, *B*, and *C* are the same, and the causal probabilities of causal relations are the same as well, then if there are alarms from *A*, *B*, and *C*, there would be no way to tell which of *A*, *B*, and *C* is the first object which had a problem.
- The fact of that there is no alarm received from an object is not considered in correlation, either.

There are some assumptions for Brunet's algorithm:

- If the timestamp information is filled by the object sending the alarm, the networks are synchronized; or the drift of clock on each object is known.
- If one uses the time when the alarm is received as its timestamp, the network latency from monitored object to the management system is either consistent or is much shorter than the time for a fault to cause another fault. Or the latency is known from each monitored object to the management system.
- The occurrences of lost alarms are rare.
- The propagation will take a breadth-first fashion rather than depth-first fashion and the propagation along different paths happen at the similar speed.

In Brunet's model, a causal model is represented as a directed graph, called *causality graph*, which is very similar to what is in Katzela's model. The differences are:

- The directed graph is called causality graph (dependency graph in Katzela's model).
- The direction of edges in the directed graph is reverse.
- The weight for an edge is $w_{ij} = -\log(p_{ij})$, where p_{ij} simply represents the conditional probability $p_{ij} = P(v_i \text{ fails} \mid v_j \text{ fails})$, while in Katzela's model, $w_{ij} = p_{ij}$.
- There is no special case, i.e., in Katzela's model, an undirected dependency graph can arise when the set D contains both edges (e_i, e_j) and (e_j, e_i) with $p_{ij} = p_{ji}$, while in Brunet's model, it will still remain as a directed graph.

- An alarm sent by an object is a message that contains two fields: the identity of the object generating the alarm, and the timestamp of the alarm, while in Katzela's model, it only contains one field: the identity of the object generating the alarm.
- There are no probabilities for vertices, so there is no way to specify the probabilities of an object fails, although it is easy to get around this drawback by adding probabilities to the dependency weight for each edge originating a vertex.

In other words, each directed edge in the causality graph represents causality: a directed edge $v_i v_j$ represents that a failure of object v_i has for possible cause a failure of object v_j . Each directed edge $v_i v_j$ is associated with a weight $w_{ij} = -\log(p_{ij})$, where p_{ij} simply represents the conditional probability $p_{ij} = P(v_i \text{ fails} \mid v_j \text{ fails})$.

Some definitions are in order before description of the algorithm. A *dependency chain* is defined as a causal chain in G where the orientation of its edges is reversed.

A *tree* is an undirected graph that contains no cycle (or loop). A *directed tree* is defined to be an undirected tree with a special vertex called the *root* where all the edges are oriented in the direction outward from the root (and therefore there is a directed path from the root of the tree to each vertex in the directed tree). A *reversed tree* is defined to be a directed tree where all the edges are reversed (and therefore there is a unique directed path from each vertex to the root of the tree). The root of the reversed tree is called the *sink* of the reversed tree. Figure 4.6(a) shows a directed tree and Figure 4.6(b) shows a reversed tree.

Figure 4.6 (a) Directed tree and (b) reversed tree

With those hypotheses and definitions, the *Temporal Alarm Correlation Optimization Problem (TACOP)* is expressed as follows. Given a causality graph G , find a vertex v and a reversed tree T in G such that:

- 1 v is the sink of T ;
- 2 T contains all the alarms;
- 3 The timestamp of T are consistent, that is, the timestamps are decreasing in a traversal of any path in T ;
- 4 The sum of the number of non-alarm vertices in each path in T from the sink (excluding it) to an alarm is minimum.
- 5 Under the assumption that conditions 1-4 are met, it is also imposed that the sum of the weights of all the paths from the sink to each alarm in T is minimized.

We now explain each restriction on the feasible space of solutions.

It is important to notice here that there are actually two optimization problems: the first one finds all solutions to conditions 1 to 4, and the second one finds the solution out of this set that satisfies condition 5.

The reason for which Brunet looks for a reversed tree containing all the alarms appears naturally: the reversed tree reflects a causal cascade of events that include the generated alarms. The sink of the reversed tree represents the primary fault that caused the cascade of faults following its occurrence.

Conditions 4 and 5 reflect that the author wants to obtain causal chains that are as certain as possible. The different dependency chains that lead from a common candidate cause to each alarm will then contribute to make the sink candidate the best possible choice.

Condition 4 represents optimality for the presence of non-alarms vertices in the causal chains. It is important to note that condition 4 reflects the main objective function in this optimality problem: find a solution that first satisfies 1-4, and then satisfies 5. Note that even in the case of the exact causal tree representing the real chain of events, it is possible that the tree is not composed of alarms only. It is assumed here indeed that the probability that some alarms are missing is non-zero. However, it is claimed that this probability must be small, as one expects to receive the maximum number of alarms from the cascade of faults. It should be clear that the determination of the possible primary causes depends on the number of missing alarms on the number of alarms generated. The greater this number, the less information one has and the more difficult the alarm correlation will be.

Note that according to condition 4, a missing alarm that is common to many paths will be counted many times in the objective function (number of non-alarm vertices), and so the weight of those paths will reduce the optimality of the solution. Because such a missing alarm implies many independent events (the events implied by this vertex according to those paths), it is reasonable to put a higher weight on it. Note also that because Brunet makes the sum of the length of the paths according to the missing alarms rather than simply counting the alarms in the tree, this objective function favors paths that have missing alarms around the origin rather than around termination. This characteristics of the objective function makes sense, since a dependency chain has more certainty if a missing vertex is at the end than at the beginning. It is important to note that the sink is never counted more than once as a missing alarm. The explanation is very simple: one missing alarm as the sink is certainly allowed but two or more at the top would lead to less solid conclusion.

The fact that the alarms received are taken into account is one important difference between this approach and the Katzela-Schwartz algorithm.

Condition 5 reflect the fact that solutions with long causal chains of low dependency weight should be avoided. In the type of situations to be addressed, the propagation of the faults is assumed to behave in a “breadth-first” expansion rather than “depth-first” expansion.

The *TACOP* problem has been proven as NP-complete. So Brunet showed that a solution to a derived problem, called *TACOP2*, is a solution to *TACOP*.

TACOP2: Find a vertex v and a set of directed paths $P = \{Pa_i \mid a_i \text{ is an alarm}\}$ in the causality graph G such that:

- 1 v is the destination of all paths in P ;
- 2 Each alarm a_i is the origin of one path Pa_i of P ;
- 3 Each path is minimum path with respect to the non-alarm metric (i.e., the number of non-alarms in Pa_i is minimum over all the directed paths in G from a_i to v);
- 4 For all paths that satisfy conditions 1-3, it is also imposed that each Pa_i is a minimum path with respect to the dependency metric.

4.3.2 Algorithm

Now let's look at the algorithm for *TACOP2*.

Brunet proposes a heuristic algorithm that will find a feasible solution to *TACOP2* under constraints 1, 2, and 3, and is a solution or an approximation under conditions 4. The next paragraphs are ordered so as to discuss the successive steps of the algorithm.

It should first be mentioned that it is assumed that the causality graph G is implemented as an array of adjacency lists. Those are the linked list of neighbors for each vertex of the graph, and the set of all those linked lists is grouped in an array.

Preprocessing phase: Computation of the Path Repository and the Transitive Closure

The following paragraphs constitute a brief discussion of the preprocessing phase to the actual diagnosis process. This phase is not a part of the diagnostic process *per se*, but its importance is capital.

The essence of the algorithm resides in a selection of possible causality chains from each of the received alarms to a possible candidate cause. In order to perform a selection of causality chains, it is necessary to have *a priori* a list of all the possible paths in the causality graph from vertex, their termination, together with their length and their dependency weight. It is easy to show that this pre-processing activity has exponential complexity. Since alarm correlation must be performed rapidly, such a process must be done *prior to* any diagnosis activity. The resulting list of paths is called the *path repository* (of the causality graph). This repository is essentially an array containing all the possible directed paths in the causality graph. Each element of the array contains information about the paths associated with this array element, but contains also the path itself (which is represented by an array of pointers, each pointer containing the address of a vertex of the path). The reason an array rather than a linked list with pointers is used to represent the paths, is that it can be useful for later analysis to access a vertex of the path directly, rather than traversing a linked list up to that vertex.

The path repository is created by traversing all the paths in the causality graph and by recording each path in an element of the path repository array. This can be achieved by taking all ordered pairs of vertices in the causality graph, and then by finding all paths from the first vertex of the couple to the second vertex of the couple of vertices chosen. Finding all such paths can be done by an exhaustive traversal of the graph from the first vertex until the second vertex

is reached. It is at the moment of this discovery of the paths that one can better get information about the path, like the length of the path, its dependency weight, its origin and destination, etc.

Given any two vertices in the graph, the algorithm requires a more condensed source of information about the paths between those two vertices. For this purpose, an auxiliary graph, the *transitive closure* $\overline{G}$ of G , is constructed in the following way: the vertices of $\overline{G}$ are exactly the vertices of G , and there is an edge $v_i v_j$ in $\overline{G}$ if and only if there is a directed path P between two vertices v_i and v_j in G . Notice that $\overline{G}$ may contain multiple edges between two given vertices: there are as many directed edges between two vertices in $\overline{G}$ as there are directed paths between the two same vertices in G . Each edge in $\overline{G}$ is assigned the weights relative to its corresponding path in G : in this case, the graph-distance of the path and the dependency weight are assigned. The set of neighbors of any vertex of the transitive closure are ordered in an increasing value of their minimum distance from this vertex. For example, suppose a vertex u has for neighbors in $\overline{G}$ v and w , and suppose that they are at distance of 3 and 7 respectively from u (the minimum length of a directed path from u to v is 3, and the minimum length of a directed path from u to w is 7). Then, in the neighborhood list of u , w will come after v . Also, the edges between any couple of vertices are ordered by increasing value of the length of the path in G they represent, and for a given value of length, they are then ordered by increasing value of the dependency weight of the paths they represent in G . In other words, given two vertices u and v in the transitive closure, their edges are organized by lexicographic order relative to the couple (*length, dependency weight*). This suggests that the set of edges between

two given vertices u and v can be represented by a double linked list attached to the element v in the linked list of the neighbors of u .

For each edge in $\overline{G}$, the algorithm needs to access the corresponding directed path in G . Each path in the path repository being in bijection to its edge in the transitive closure of $\overline{G}$, this bijection (one-to-one mapping) is represented by a couple of pointers between the path of the path repository and its edge in $\overline{G}$.

Both the transitive closure and the path repository of a causality graph can be large, but they are fixed data structures in the main database, and they do not need to be updated during the alarm correlation process. They can actually be stored in a file.

We can now describe the main algorithm itself. In the input of the algorithm, besides the causality graph G just described above and its auxiliary graphs, two parameters are needed: a *distance bound parameter* δ_d and a (*dependency*) *weight bound parameter* δ_w . The alarms are taken from the set of alarms received within a time window t .

- *Input:*
 - causality graph G ;
 - its transitive closure $\overline{G}$;
 - its path repository $PathsRep(G)$;
 - the set of alarms received during a window of time t ;

- a *distance bound parameter* δ_d ;
- a *(dependency) weight bound parameter* δ_w .
- *Output*: a reversed tree solution T_{sol} to TACOP2.

Phase I: Processing of the alarms

As the alarms are received, the first step is to order them according to their timestamps. It is assumed that the alarms are grouped in a linked list. Their ordering can be done very easily and efficiently in polynomial time $O(N \log N)$, where $N = |G|$, and will help in later analysis. Subsequent alarms are also inserted in this list according to their timestamps.

Phase II: Constructing of the Cause-Alarm Bipartite Graph

A *Bipartite Graph* is a graph whose vertex set can be partitioned into two subsets X and Y , and each edge in the graph has one end in X and one end in Y . In this case, the *Cause-Alarm Bipartite Graph* G' has its vertex set partitioned into the set of received alarms A and the set of possible causes C .

The edges are always oriented from a cause towards an alarm. They represent a dependency path that links an alarm to a cause. Therefore, they correspond to an edge of the transitive closure of the causality graph, but with a reverse direction. As for $\overline{G}$, multiple edges are possible in G' . Figure 4.7 shows an example of a cause-alarm bipartite graph.

Figure 4.7 Cause-Alarm graph

The main point in the construction of G' is to optimally choose the candidate causes (the vertices of C) and the causality paths from the received alarms to these causes. This stage of the algorithm is extremely important, as the number of possible candidates chosen will directly influence the complexity of the algorithm. If one chooses all the possible destinations of causality chains from each alarm, he/she could end up with exponential complexity. The candidates need to be chosen extremely carefully. The distance bound (respectively dependency weight bound) represents the greatest distance or length (respectively dependency weight) within which the paths from each alarm should be taken. In other words, only paths that have length and dependency weight lower than K_d and K_w respectively are searched and analyzed. Two methods are proposed: a general method (method *A*) where the values of $K_d = \delta_d$ and $K_w = \delta_w$ are used as bounds directly. Another method (method *B*) scale these two parameters for each alarm according to the position of the alarm in the sorted alarm vector.

Method A: For this purpose, the search will be bounded within the values K_d and K_w : for each alarm a_i received, we find all the vertices v that have at least one path from a_i to v with length lower than δ_d and dependency weight lower than δ_w .

Method B: This second method is similar to method A, except that δ_d and δ_w are scaled for each alarm according to its position in the alarm vector. More specifically, for each alarm a_i received, we find vertices v whose distance from a_i is within a certain function value $K_d(\delta_d, a_i)$ and whose dependency weight is within a certain function value $K_w(\delta_w, a_i)$. One option in the algorithm is to make the distance bound an actual function of the timestamp of the alarm: for example, for a_1 , it is assumed that the cause should be within a distance at most 2 from it. For the latest alarm, it is assumed a distance bound that should be bigger. In general, a distance bound $K_d(\delta_d, a_i) = \delta_d (1 + 2\log_2 i)$ is a good guess. For the dependency weight bound, the value depends on the degree of certainty needed, depending on the alarm. For example, if it is known that usually a chain that leads to the emission of a certain alarm a_i is near certainty, a bound value $K_w(\delta_w, a_i)$ that is close to 0 will be used.

In both methods, the candidate causes are selected from the transitive closure. The reader remembers that the neighbors of a vertex have been ordered according to their distances (shortest distance), and their edges according to their distances but also to their dependency weight (within a common distance value) in a lexicographic manner. From this ordering, it becomes easy to select all vertices that are within the assigned bounds by traversing the neighbors of an alarm in the transitive closure up to the distance bound and the edges up to the two bounds. When a candidate cause is added to the cause-alarm graph, a reversed copy of the

transitive closure edges between the alarm and the cause whose value is lower than K_d and K_w , is also added. Thus, the Phase II Algorithm is:

Phase II Algorithm

For (each alarm) do

While (there is a neighbor to alarm at distance $< K_d$) do

*While (there is an edge from alarm to cause in G of length $< K_d$ and of
dependency weight $< K_w$) do*

Add neighbor to G' ;

Add the reversed edge to G' ;

End While

End While

End For

The cause-alarm graph is identical to Yemini's correlation graph, except for the fact that here Brunet adds all the edges representing possible causality chains in G between a given cause and alarm, within the bounds K_d and K_w . This graph represents all the causes that can explain some alarms (or all) within a certain approximation, and the possible explanation chains to the alarms. While the causality graph, its transitive closure and the path repository are fixed data that do not depend on that alarms, the cause-alarm graph is constructed with each alarm window received.

The cause-alarm graph will be the main instrument to determine the best candidates as primary causes of the alarms received. The next step in the algorithm is to consider the connected components of the cause-alarm graph. The alarms of a connected component of the bipartite graph should be investigated together, since this “proximity” within a connected sub-graph of any two alarms suggests the likelihood of a unique cause to those alarms. Such a connected component of the cause-alarm graph constitutes exactly what Katzela and Schwartz call a cluster of alarms. The determination of the clusters, within the construction of cause-alarm graph, is easy. Indeed, the alarms are processed in order and each alarm is checked to see if a possible cause to it in the graph is adjacent to some other alarms. The lists of those adjacent alarms and causes are made, and each of those lists becomes the set of vertices of a cluster. This can be done in polynomial time.

Each cluster is processed one at a time, and one possible primary cause is considered per cluster. This does not mean that there has to be one primary cause in a cluster that can explain all the alarms in the cluster. For the rest of the explanation of the algorithm, it is assumed that there is only one cluster in the cause-alarm graph, since the multiple clusters case is simply a repetition of the single cluster case.

The next step in Phase II of the algorithm consists of partitioning the causes into two sets: the single cause list and the multiple cause list. The single cause list consists of all causes that are adjacent to each alarm. Those have clear potential to be the primary causes to all alarms. The multiple cause list is therefore the rest of the causes. Those will be examined later. This single cause list is constructed as a linked list where the best candidate of the list is always put

at the head of the list. Then, each cause of the cause-alarm graph is simply scanned in turn: if the cause is adjacent to each alarm and its timestamp (if it has one) is lower than or equal to each timestamp of the rest of the alarms, then it will be inserted in the single cause list. If its timestamp (if it has one) is lower than or equal to each timestamp of the rest of the alarms but does not explain all the alarms, this cause will be put in the multiple cause list. Otherwise, the cause will be deleted.

It is important to notice that the edges of the cause-alarm graph are ordered, for each pair of vertices, in increasing value order.

Phase III: Pruning the cause-alarm graph

The next phase of the algorithm consists of “pruning” the cause-alarm graph. As we have accumulated all possible candidate causes and causal chains within the heuristic parameters of distance and dependency weight, we want to eliminate as many of those candidates as possible in order to pinpoint the primary cause of all alarms. For this, the candidate causal paths in G are scanned to see if there is some inconsistency in the timestamps of the alarms and causes. If, during the traversal of the path from the alarm to the cause, the sequence of timestamps is not strictly decreasing, this causal path has to be rejected, since it implies that the cause came after the effect. If an edge in the cause-alarm graph is the only edge between a candidate for primary cause and an alarm, invalidating this way the quality of this cause to be a global primary cause, the cause is removed from the global cause list and the algorithm moves to the next cause in this list.

In order to get a solution that respects condition 3 of *TACOP2*, the non-alarm measure of the paths has to be recorded during traversal of them. Later, after having pruned the paths with respect to timestamps, the remaining paths will be re-ordered according to their non-alarm measure, their weight and their length.

In a conclusion, Brunet's algorithm has a few good features:

- Split the real time calculation and non-real time calculation. This would make the real-time diagnosis much faster since a large portion of the calculation in the correlation, which is the same for each correlation for every alarm, is done "off-line".
- Dealing with the temporal relationships.
- Handles loops in the causal model.
- Less sensitive to probabilities assigned to edges.

4.4 Comparisons of alarm correlation algorithms

Katzela's algorithm and Brunet's algorithm will be compared in this section. To make the meaning of the categories in which two algorithms are compared clear, the explanation of the categories is in order.

4.4.1 Scalability

There are four aspects of scalability: accuracy, speed, memory requirements, and distributed.

- *Accuracy* is how often the root cause(s) as a result of the algorithms matches the actual root cause(s). Usually the more complex the causal model is, the less accurate the calculated root cause(s) will be.
- *Speed* is how fast the algorithm finds the root cause(s). Usually the more complex the causal model is, the longer it will take for an algorithm to find the root cause(s).
- *Memory requirements* is how much memory the algorithm needs to find the root cause(s). Usually the more complex the causal model is, the more memory an algorithm will need to find the root cause.

Notice that each of these three aspects is mutually exclusive to other two. For example, it would either take longer or need more memory if a higher degree of accuracy is required.

- *Distributed*: For large causal models, it is desirable that the alarm correlation can be done in a distributed fashion. But it may be very difficult to make some algorithms work in a distributed manner.

4.4.2 Evaluation of Brunet's algorithm

- *Accuracy*: The bound setting plays a big role in accuracy of the algorithm. If the probability of a path i is P_i , U is the set of paths which contain all the alarms present, and the timestamps are consistent, and are within weight and length bounds. B is the set of paths

which contain all the alarms present, and the timestamps are consistent, but are beyond either the weight or length bound, or both. Then the accuracy of the correlation, P_a , which is expressed as the probability of that the algorithm finds the right cause, can be expressed in the following formula:

$$P_a = \frac{\sum_{i \in U} P_i}{\sum_{i \in B} P_i + \sum_{i \in U} P_i}$$

So the looser the bounds are (i.e., the weight bound is larger, and/or the length bound is longer), the more accurate the correlation will be.

Please note that the algorithm does not count for aggregate probabilities. E.g., if there is only one alarm, the algorithm will favor a fault to which there is a path from the alarm vertex and the path weight is the smallest, even though there may be another fault to which there are multiple paths from the alarm vertex and the aggregate probability of these paths is larger than that for the former fault vertex. It would be fine that all paths from the same alarm to the same fault are mutually exclusive to each other. However, it is not always true that these paths are mutually exclusive to each other.

- *Speed:* Let's see the upper bounds of the computational complexity of Brunet's algorithm. First we consider the case of a complete directed graph (a graph in which there is a directed edge from every vertex to every other vertex). Assume there are N vertices. There will be

$N(N-1)$ couples of source vertex and target vertex and about $e(N-2)!$ paths from each vertex to each other vertex. So the total number of paths is $e \cdot N!$. So the computational complexity of path repository construction is $O(N!)$, the complexity of transitive closure construction is $O(N! \log(N!))$ because it needs to do sorting. If we receive L alarms, we will have L vertices for alarms in the cause-alarm bipartite graph and $eL(N-2)!$ paths so the complexity of construction of the cause-alarm bipartite graph is $O(N!)$. The complexity of the algorithm is thus $O(N! \log(N!))$.

But in practice, a causal graph will not be a complete directed graph. Now assume that each vertex has a maximum of M egress edges. The maximum number of paths from one vertex to another is close to $M^{(N-M-1)}M!$, the total number of paths has a maximum of $N(N-1)M^{(N-M-1)}M!$. Usually M does not increase when N increases, so the complexity of path repository construction is $O(N^2M^N)$, the complexity of the transitive closure construction is $O(N^2M^N \log(M^N)) = O(N^3M^N)$, the complexity of the cause-alarm bipartite graph is LM^N , and the complexity of the algorithm has an upper bound $O(N^3M^N)$.

So in the worst case (we are not considering complete graphs), Brunet's algorithm has an exponential complexity.

- *Memory requirements:* Because in the worst case, the total number of paths grows exponentially as shown in the previous item, and Brunet's algorithm requires each path be read into the memory to do correlation, the memory requirements of Brunet's algorithm will grow exponentially in the worst case.
- *Distributed:* The construction of the path repository can be distributed. That is, finding all paths between two vertices is independent of finding all paths between other two vertices.

Actually the correlation part can be distributed, too, since checking how many alarms are present in a path and the timestamp consistency of two paths can be done independently from those for other paths.

4.4.3 Performance improvements to Brunet's algorithm

We have already shown that Brunet's algorithm could have an exponential complexity. So it is desirable to improve the algorithm. We found a way to improve the algorithm: application of bounds during path repository construction. A path will not be saved in the path repository if it is outside the weight and/or length bounds. Moreover, we will not search further if the sub-path is already outside the bounds. This saves both time and memory, i.e., makes the algorithm much faster and requires much less memory.

Now let's see how speed and memory requirements will be improved. Assume that now the maximum length of paths is K , which will not be larger than the length bound. The maximum of the total number of paths will be $M^K N(N-1)$, the complexity of the preprocessing (path repository and transitive closure construction) and (since the complexity of Phase I, II, and III is only $O(LM^K)$) the algorithm will be $O(N^2)$, if the M and K are not increasing when N increases, or even if M increases, but not worse than $\log N$, the complexity will have an upper bound of $O(N^3(\log N)^K)$, in either case, we can avoid an exponential complexity. The memory requirements will increase in an order of N^2 . The distributed nature of the algorithm will not be compromised. The accuracy of the algorithm will not be worse than that of Brunet's algorithm if we choose bounds which are larger than those used in Phase II. The drawback of this approach, however, is that we can only loosen the bounds in Phase II to the bounds set in

the path repository construction. If we need to loosen them more, we have to calculate the path repository again.

Please note that the complexity of Brunet's or our improvements depend on the nature of the causal graphs (like how many loops there are, etc.). But by introducing our improvements, we can avoid an exponential complexity, which may happen to Brunet's algorithm. Our improvements will have at least the same complexity and memory requirements as those of Brunet's algorithm, if not better.

Another improvement is incremental path repository construction. In Brunet's algorithm, the path repository construction is done by doing a depth-first search of all egress edges from the source vertex. An edge will be scanned as many times as the number of paths which is consisted of the edge and other edges, if any. The improvement is to scan each edge only once and for each vertex keep track of all paths targeted at it and originated from it. When a new edge is scanned, we append the edge to each of the paths targeted at the source vertex to form a new path, prefix the edge to each of the paths originated from the target vertex to form a new path, and finally, for each pair of path targeted at the source vertex and path originated from the target vertex, we bridge them together with the new edge to form a new path. In each case, we will check if the newly formed path is loop-free before saving it to the path repository. This makes the path repository construction much faster, the drawback is that it is less likely it can be distributed, and it requires more memory (but it is still better than that of Brunet's algorithm, the memory still increases at an order of N^2).

4.4.4 Comparisons

Now let's compare the two algorithms and the improvements we have done. The results from the experiments are listed in Section 5.2.3 on page 123. Please note in Section 5.2.3 we only compare Brunet's algorithm and our improvements. The reason is that Katzela's algorithm can not be applied to our causal model because it can not handle loops.

- *Accuracy:* We have already shown that Katzela's algorithm can not produce accurate diagnosis when there are loops in the causal graph. Brunet's algorithm fixes the problem and includes timestamp to help diagnosis, thus the accuracy is better, while our improvements have the same accuracy of Brunet's.
- *Speed:* The complexity of Katzela's algorithm is $O(N^3)$. Thus it is better than Brunet's. Our improvements is between $O(N^2)$ and $O(N^3(\log N)^k)$ so it can be slightly better or slightly worse than Katzela's.
- *Memory requirements:* Memory requirements of Katzela's algorithm grow in an order of $O(N)$ for the worst case (we omitted the memory required to store the causal graph, which increases in the same order for all algorithms and has $O(N^2)$ as the upper bound). So it is best among three while Brunet's algorithm is the worst.
- *Distributed:* Both Katzela and Brunet have demonstrated a way to make their algorithm distributed, so both are very good at this category. Our first improvement does not compromise it, while the second one does compromise the distributiveness.

So the comparisons can be summarized in the following table:

Table 4.1 Comparisons of alarm correlation algorithms

Items to be compared	Katzela's algorithm	Brunet's algorithm (without our improvements)	Brunet's algorithm (with our improvements)
Scalability (accuracy)	moderate	good	good
Scalability (speed)	good	moderate	good
Scalability (memory)	excellent	moderate	good
Distributable	good	good	moderate (good if we include the first improvement only)

4.5 Shortcomings

There are some common shortcomings in the algorithms described above, e.g.:

- No compound causal relationships (e.g., AND) can be accommodated. Thus, it is almost impossible to describe such relations as “if fault 1 and fault 2 happen (at the same time), fault 3 will happen”.
- No delay and/or times can be specified. Some causal effect happens only after certain period of time after the cause happens, sometimes another fault/alarm comes up only after the causing fault happens several times.
- Do not allow the probability of an alarm will be sent in event of a fault be specified.

The Brunet's algorithm (even with our improvements) and model have two more shortcomings:

- Can be used for fault localization but can not be used for fault identification.
- Probabilities of vertices can not be included in correlation.

We will explain the last three drawbacks and how the problem can be solved in Section 5.2.1 on page 109.

Chapter 5 Alarm Correlation in ATM Networks

In this chapter, we first describe the way in which important faults/alarms are propagating in ATM networks, include propagation between switches. Then we discuss how Brunet's algorithm can be extended and applied to the causal model we have for ATM networks. At last, we evaluate the performance and scalability of Brunet's algorithm, and compare these to those of Katzela's algorithm.

5.1 Alarms and their propagation in ATM networks

Before illustrating the alarm propagation in ATM networks, the following sub-section describes the model of ATM networks we use in this chapter.

5.1.1 Model of ATM Networks

The model of ATM networks is shown in Figure 5.1 on page 88. We assume there is an ATM layer for each port. Note that this does not mean it requires the certain implementation of an ATM switch have an ATM layer for each port. Instead, what is required is that every alarm from the ATM layer contain information about which port this alarm is related to.

Figure 5.1 The model of ATM networks

We also assume that for each switch, there is only one AAL layer, which contains AAL1, AAL3/4, AAL5. Also note that this does not preclude that an implementation of a switch has a separate AAL layer for each port. Instead, this just removes the requirement that an alarm from AAL layer contain information about which port is related.

5.1.2 Important Alarm/Error Propagation

This section will illustrate important alarm/error propagation in ATM networks. The dashed lines denote propagation which are very unlikely, unlikely enough to be ignored, while the forked lines denote possible propagation from an alarm to one of several other alarms. The faults marked with a star (*) propagate between connected ports in two switches, while the faults marked with a pound sign (#) propagate between ports in a switch. These will be examined in more details in later sections.

Please note that some faults/alarms illustrated in the following sections have different sources including SNMP traps, alarms generated by performance thresholding, and OAM cells. Section 5.1.3 on page 105 will examine this in more details.

The following figures show the important alarm propagation in:

- Physical Layer (Figure 5.2)
- ATM Layer (Figure 5.3)
- AAL1 Layer (Figure 5.4)
- AAL3/4 Layer (Figure 5.5)
- AAL5 Layer (Figure 5.6)
- SAAL (Signalling AAL) Layer (Figure 5.7)

Figure 5.2 Alarm propagation in ATM networks (Physical Layer)

Figure 5.3 Alarm propagation in ATM networks (ATM layer)

Figure 5.4 Alarm propagation in ATM networks (AAL1)

Figure 5.5 Alarm propagation in ATM networks (AAL3/4)

Figure 5.6 Alarm propagation in ATM networks (AAL5)

Figure 5.7 Alarm propagation in ATM networks (SAAL)

5.1.2.1 Alarm propagation between switches

The important alarm propagation between switches (connected ports) is similar to what is illustrated above, the only difference is that it is only applicable to some faults shown in Figure 5.8 on page 96. These faults/alarms are first propagated horizontally to neighboring nodes, then upward. An example is shown in Figure 5.9 on page 97. As shown in the figure, the fault “*Protocol Error*” is first propagated from a port in *switch 1* to its connected port in *switch 2*, cause the same fault (“*Protocol Error*”), then “*Protocol Error*” in the port of the *switch 2* will cause other faults/alarms such as *LOS*, *LOF*, etc. So there is an indirect causal relationship between “*Protocol Error*” in *switch 1* and “*Alarms (LOS, LOF, etc.)*” in *switch 2*.

Also note that “*Protocol Error*” in *switch 1* will cause “*Protocol Error*” in *switch 2*, and vice versa. In other words, there is a loop in the causal model, between these two faults in different switches. The same holds true for other faults shown in Figure 5.8 on page 96.

Figure 5.8 Important problems propagated between connected ports of two switches

Figure 5.9 An example of alarm propagation between neighboring switches

5.1.2.2 Alarm propagation between ports in a switch

The important alarm propagation between ports in the same switch is shown in Figure 5.10. As illustrated in the figure, whenever there is a cell with wrong *CLP* (Cell Loss Priority), the problem will be propagated either to upper layers (so other problems in upper layers would occur as shown in previous figures), or to other ports (so the same problem will appear on another port). Notice that this propagation is mutual, so there will be loops in the causal model.

The other six problems shown in the figure have the same propagation pattern between ports in a switch.

Figure 5.10 Alarm propagation between ports in the same switch

An example of the alarm propagation between ports in the same switch is shown in Figure 5.11 on page 99. As shown in the figure, there are also loops. And again, some faults are propagated horizontally first, then upward.

Figure 5.11 An example of alarm propagation between ports in a switch

5.1.2.3 Combined causal model

So far, we have described fault/alarm propagation between layers in a switch, between two connected switches, and between ports in a switch. In this section we describe them in a combined context. Figure 5.12 on page 102 shows the all three types of propagation together in a network composed of three switches. Note not all propagation is shown in the figure because of space constraints. Also because of the same reason, we use a number instead of a description in the vertex. The number is comprised of two parts: the first one-digit part

designates which port and/or switch a vertex is related (the mappings are shown in Table 5.1 on page 100), while the second two-digit part specifies which fault/alarm is (the mappings are shown in Table 5.2 on page 100).

Table 5.1 The mappings between the first digit of vertices and switches/ports

The first digit of vertices	Switches/ports
1	Port 1 of switch 1
2	Port 2 of switch 1
3	Port 1 of switch 2
4	Port 1 of switch 3
5	Higher layers (AAL and over) of switch 1
6	Higher layers (AAL and over) of switch 2
7	Higher layers (AAL and over) of switch 3

Table 5.2 The mappings between the last two digits of vertices and faults/alarms

The last two digits of vertices	Faults/alarms
01	Bit Error
02	Physical Layer Protocol Error
03	Link Down
04	Cell Payload Bit Error
05	Cell Header Bit Error
06	Overhead Bit Error
07	BIP Error
08	Correctable HEC Error
09	Out of Cell Delineation
10	Uncorrectable HEC Error
11	Alarms (LOS, LOF, etc.)

Table 5.2 The mappings between the last two digits of vertices and faults/alarms

The last two digits of vertices	Faults/alarms
12	Cells Dropped in Physical Layer
14	ATM Layer Protocol Error
19	Wrong CLP
22	Cells discarded in ATM Layer
23	Cells discarded
35	Correctable SAR-PDU Header Error

As shown in the figure, the fault “*Cell Payload Bit Error*” can propagate from *port 1* of *switch 2* (vertex 304), to *port 1* of *switch 1* (vertex 104) because there is an ATM link between these two ports. Then the same fault will propagate to *port 2* of *switch 1* (vertex 204), and at last to *port 1* of *switch 3* (vertex 404). The same fault can propagate in the reversed direction. Again, since we assume there is only one AAL layer for each switch, the vertices for AAL layers and upper layers has only one for each fault/alarm in each switch. This is shown in the figure as vertex 535 is the only alarm vertex for “*Correctable SAR-PDU Header Error*” for *switch 1*, even though there are two ports in the switch 1. There is a causal relation between vertex 104, which is for *port 1*, and vertex 535. There is also a causal relation between vertex 204, which is for *port 2*, and vertex 535. In other words, the two ports share the same vertices for AAL and upper layers.

Figure 5.12 A sample causal model

5.1.2.4 Probabilities

It is quite difficult to get accurate probabilities for some faults and some causal relationships. This is part of the reason why the alarm correlation is very difficult. However, some probabilities (like bit error rate of fiber), especially relative probabilities, can be calculated. An example is probabilities of the causal relations from “*Bit Error*” to *CPBE*, *CHBE*, and *OBE* (see Figure 5.9 on page 97 for these acronyms) in SONET environment. Since there are 3 rows of overhead and 87 rows of payload out of 90 rows of an STS1 frame, and the ATM cell header is 5 bytes, the payload is 48 bytes, and the total length of an ATM cell is 53 bytes. The media of ATM networks has a very low bit error rate so the probability of two or more bit errors in a data block is very low (like at a bit error rate of 10^{-6} , the probability of 2 or more bit errors in STS 1 payload is only 0.32% of that of single bit error), and so we do not need to consider multiple bit errors. This is how we did in the experiments, it makes testing of causal model easier while it does not impact the accuracy of the result. So if the bit errors

happen at random locations in an STS1 frame, the ratio of the probability of bit error happening in STS1 overhead to that of bit error happening in STS1 payload is very close to 3:87, we have:

1

$$\frac{P_{CHBE}}{P_{OBE}} \approx \frac{5}{53} \times \frac{87}{3}$$

2

$$\frac{P_{CPBE}}{P_{OBE}} \approx \frac{48}{53} \times \frac{87}{3}$$

3

$$\frac{P_{CHBE}}{P_{CPBE}} \approx \frac{5}{48}$$

Probabilities of faults like “*Protocol Error*” are hard to estimate. The probabilities of causal relations like from “*Bad CAC*” to “*Buffer full*” are hard to estimate, too. To make the matter even more complicated, some probabilities may change over time. E.g., bit error rate decreases when the quality of medium (fiber, etc.) products is getting improved, protocol error will getting more and more unlikely when the technology matures and/or vendors are understanding the standards better.

5.1.3 The sources of the alarms

The alarms described in earlier sections have different sources/origins. Among others, the most important ones are:

- SNMP traps
- Alarms generated by performance thresholding by an RMON agent and/or mid-level managers, as described in Chapter 3.
- Alarms sent as OAM fault management cells, also as described in Chapter 3.

Table 5.3 shows the sources of the alarms.

Table 5.3 The sources of alarms

Alarms	MIB Object(s)	Trap/Notification	OAM
Bit Error		atmfM4HwTimingProb Alarm	
Link down		Link down, atmfM4HwReceiverFail Alarm, atmfM4HwXmitterFailAlarm	
BIP Error	sonetSectionCurrentCVs, sonetLineCurrentCVs, sonetPathCurrentCVs, sonetVTCurrentCVs, sonetSectionBIPs		
Correctable HEC Error	sonetAtmCorrectableHCSSs		
Out of Cell Delineation (OCD)	atmInterfaceOCDEvents, atmInterfaceTCAlarmState	atmfM4IfLcdAlarm	
Uncorrectable HEC Error	ifInErrors, sonetAtmUncorrectableHCSSs		

Table 5.3 The sources of alarms

Alarms	MIB Object(s)	Trap/Notification	OAM
Alarms (LOS, LOF, etc.)	sonetSectionLOSSs, sonetSectionLOFs, etc.	atmfM4IfAisAlarm, atmfM4IfLofAlarm, atmfM4IfLosAlarm, atmfM4IfXmissionErrAl arm	AIS, RDI
Cells Dropped in Physical Layer	ifInErrors, atmfAtmStatsDroppedReceiv edCells		
Link Failure	sonetLineCurrentUASs, sonetPathCurrentUASs, sonetVTCurrentUASs		
Wrong VPI/VCI	atmOutOfRangeVPis, atmUnconnectedVPis, atmOutOfRangeVCis, atmUnconnectedVCis (fore) An entry present in atmfM4CellProtoErrorTable with unassignedVpiVciValue or outOfRangeVpiVciValue for atmfM4CellProtoErrorReason		
Wrong PT		atmfM4IfPayloadMisma tchAlarm	
Traffic Disagreement or wrong policing	atmfM4VpUpcNpcCurrDiscar dedCells, atmfM4VpUpcNpcCurrDiscar dedClp0, atmfM4VpUpcNpcCurrPasse dCells, atmfM4VpUpcNpcCurrPasse dClp0, atmfM4VcUpcNpcCurrDiscar dedCells, atmfM4VcUpcNpcCurrDiscar dedClp0, atmfM4VcUpcNpcCurrPasse dCells, atmfM4VcUpcNpcCurrPasse dClp0		

Table 5.3 The sources of alarms

Alarms	MIB Object(s)	Trap/Notification	OAM
Buffer full		atmfM4HwCongestionAlarm	
Queue too long		atmfM4HwCongestionAlarm	
Correctable SAR-PDU Header Error (AAL1)	Attribute "Header Errors" of the managed entity "AAL1 Protocol Current Data" (M4 Interface Requirements and Logical MIB) [ATMF1998a]		
Uncorrectable SAR-PDU Header Error (AAL1)	The attribute "Header Errors" of the managed entity "AAL1 Protocol Current Data" (M4 Interface Requirements and Logical MIB)		
Bad Sequence Number for AAL1 PDU	The attribute "Sequence Violations" of the same entity above		
SDT Error	The attributes "STD Pointer Reframes" and "STD Pointer Parity Check Failures"		
Cells arrives untimely	The attributes "Buffer Underflows" and "Buffer Overflows"		OAM performance monitoring
Correctable SAR-PDU Header/Trailer Error (AAL 3/4)	The attribute "Sum Of Invalid SAR Field Errors" of the managed entity "AAL3/4 Protocol Current Data" (M4 Interface Requirements and Logical MIB)		
Uncorrectable SAR-PDU Header/Trailer Error (AAL 3/4)	The attributes "Sum Of Invalid SAR Field Errors" and "Sum Of Incorrect SAR Field Errors" of the managed entity "AAL3/4 Protocol Current Data" (M4 Interface Requirements and Logical MIB)		
Wrong MID/data	The attribute "BOMS EOMS Unexpected MID"		

Table 5.3 The sources of alarms

Alarms	MIB Object(s)	Trap/Notification	OAM
Bad Sequence Number for AAL3/4 PDU	The attribute "COMS EOMS Unexpected SN"		
Reassembly fails (wrong ST/SN, LI, CRC)	The attributes "Sum Of Invalid SAR Field Errors" and "Sum Of Incorrect SAR Field Errors"		
Reassembly timeout for AAL3/4 PDU	The attribute "SRI Time Outs"		
Wrong CPI	The attribute "Sum Of Invalid CS Field Errors"		
Btag, Etag mismatch	The attribute "BE Tag Mismatch"		
Wrong BAsize, length	The attribute "Length BA Size Mismatch"		
CS-PDU discarded	The attribute "Sum Of Invalid CS Field Errors" and "Sum Of Incorrect CS Field Errors"		
Wrong port/protocol	udpNoPorts/ ipInUnknownProtos		
Assembly fails or retransmission	tcpRetransSegs		
Length violation or CRC error (AAL5)	aal5VccCrcErrors and aal5VccOverSizedSDUs The attributes "CRC Violations" and "Sum Of Invalid CS Field Errors" of the managed entity "AAL5 Protocol Current Data" (M4 Interface Requirements and Logical MIB)		
Reassembly fails for AAL5 PDU	The attribute "Sum Of Invalid CS Field Errors"		
Reassembly timeout for AAL5 PDU	aal5VccSarTimeOuts The attribute "Reassembly Timer Expirations"		

Table 5.3 The sources of alarms

Alarms	MIB Object(s)	Trap/Notification	OAM
Assembly fails or retransmission for SSCOP PDU	atmSigSSCOPErrPdus The attribute "SSCOP Errored PDUs" of the managed entity "SSCOP Current Data" (M4 Interface Requirements and Logical MIB)		
Protocol anomaly/ Performance (setup time, etc) degradation	atmSigDetectUnavailRoutes, etc. (ATM Supp. MIB) The attribute "SSCOP Connection Monitoring Counter"	atmfM4HwCallEstErrAlarm	

5.2 Application of Brunet's algorithm

5.2.1 Extension of the causal model

Brunet's algorithm is selected for application of the causal model because it handles loops and temporal relationships, and it is less sensitive to probabilities assigned to edges (this feature is significant because the probabilities, as we said, are difficult to estimate). To apply it, an engine is created to generate causal graphs based on the network topology.

Before applying the algorithm, a few extensions are required to make Brunet's model practical enough to be useful for the causal model for ATM networks described above. Before describing what extensions have been done, we explain why the extensions are necessary. According to Bouloutas *et al* [Bouloutas1994], the process of fault identification in large communication networks can be divided into three steps:

- The first step is *fault detection*. Fault detection can be thought of as an on-line indication that some component of the network is malfunctioning. Usually communication devices

provide indications (in the form of alarms) when they sense a malfunction. Thus, fault detection can be accomplished using the alarms provided by the network devices.

- The second step is *fault localization*, i.e., the analysis of the alarms from the devices of the network in order to propose possible hypotheses of faults. This step is essential since, in most cases, alarms do not provide explicit detailed identification of the malfunctioning device.
- The third step is *fault identification*. The actual fault is isolated given a number of hypotheses of faults.

Again according to Bouloutas *et al* [Bouloutas1994], ideally an alarm provides the following information:

- *Who*: The name of the system reporting, or if different the name of the system experience the fault.
- *What*: The condition of the fault, i.e., the symptom of the fault.
- *Where*: A description of the position in the network where the problem occurred.
- *When*: A description of the time at which the problem occurred.
- *Why*: The cause of the problem, i.e., the nature of the fault.

However, because of lack of knowledge sometimes *Where* and *Why* are not provided, this is what we need alarm correlation to resolve. From the review of the algorithms in Chapter 4, we know in Katzela's model, the alarms only provide *Who*, while in Brunet's model, the alarms

only provide *Who* and *When*. But in the causal model for ATM networks described in previous sections, the alarms provide *Who*, *When*, and *What*.

Brunet's algorithm can be used for fault localization, but it is not designed to do fault identification, because the algorithm only tries to locate the object(s) with faults, but does not try to identify which specific fault actually was. In Brunet's model, there are only vertices representing "*objects*" which may have faults and alarms, but there is no way to specify what the faults and alarms are. So when it is required to correlate multiple alarms (which provide *What* information) from a network entity (physical layer, ATM layer, etc.) and diagnose which specific fault(s) had been there, i.e., to do fault identification, there will be no way to do correlation if each entity is represented as a vertex.

But the purpose of the causal graph for ATM networks described in previous sections is for both fault localization and fault identification.

Because fault identification would be made possible if each fault and/or alarm is represented as a vertex in the causality graph, one logical step is to convert each fault and alarm to a vertex, but then we will have the four following types of vertices (depends on whether the

vertex represents a candidate of a primary fault or not, and whether there is an alarm associated with the vertex or not):

- **Type A:** The vertex can be a candidate for the primary fault, but it will not generate an alarm. "Bit Error" is an example.
- **Type B:** The vertex can be a candidate for the primary fault, and it generates an alarm also. "Traffic Disagreement or wrong policing" is an example. This is what a vertex is in Brunet's model.
- **Type C:** The vertex is only a secondary cause, i.e., it can never be a candidate for the primary fault, but it generates an alarm. "Wrong VPI/VCI" is an example.
- **Type D:** The vertex is only a secondary cause, and it does not generate an alarm. "Cell Header Bit Error" is an example. These vertices are often included to make the causal model more understandable and thus easier to maintain.

we need to do more to prevent the algorithm from producing wrong diagnosis. Otherwise, the algorithm would yield wrong diagnosis because:

- It favors causal chains with fewer missing alarms so it will rank causal chains with more *Type A* and/or *Type D* vertices (because there will not be alarms generated for them) as less optimal.
- It would diagnose a *Type C* or *Type D* vertex, i.e, a vertex representing a secondary cause only, as a primary fault.

Another problem is that during path repository construction, it will include those paths which do not start with an alarm and/or not targeted at a primary cause in the path repository. These paths will not help alarm correlation and only take spaces.

So the extensions to Brunet's model include, but not limited to the following:

- Convert faults and alarms to vertices (objects).
- Add two more attributes to the vertices: one to specify whether it can be a primary cause and another to specify whether this vertex can generate alarms. By adding these two attributes we can (and did) modify the algorithm to prevent the algorithm from pruning out the causal chains with many vertices which are not supposed to generate alarms, and from reporting vertices only for secondary causes as primary causes.
- As said in Section 4.5 on page 85, Brunet's model does not allow to specify what the probability of a fault (that the object represented by the vertex becomes faulty) is in Brunet's model. To circumvent the problem, either the weight can be added to every edge targeted at this vertex, or the probability of a problem can be specified by introducing another "transient" causal vertex, with an edge from the problem to this new vertex having the probability of the fault as its dependency weight.
- Again as said in Section 4.5 on page 85, there is no way to specify the probability that an alarm will be sent when a fault occurs in Brunet's model. It assumes that every time the faults occurs, an alarm will be sent, but this is not true in practice, for example, very often the network element does not detect the fault and so it does not send an alarm. Specifying

this probability is made possible by introducing another vertex which serves as the alarm and assign the probability to the edge from the alarm to the fault.

5.2.2 Generation of the causal graphs

As shown in Section 5.1.2.3 on page 99, because the combined causal model is very complicated and varies when the topology changes, it would be very time-consuming and error-prone if the causal model is constructed manually, so it is desirable to generate the causal model automatically, based on the topology. An engine is created (in Perl) so a causal model can be generated, using a file describing the topology of the network and another file describing the base causal graph as inputs. This engine is referred as causal model generator in this paper. This is shown in Figure 5.13 on page 115.

Figure 5.13 The causal model generator

The syntax of the network topology description file is:

- Lines starting with “#” are comments
- A switch is specified by a line which has the following syntax:
`node <node-number> <node-name>`
- A port is specified by a line which has the following syntax:
`port <node-number|node-name> <port-number> <port-name>`

- A link between *port-name1* on switch *node-name1* and *port-name2* on switch *node-name2* is specified by a line which has the following syntax:

```
link    <node-number1|node-name1>    <port-number1|port-name1>
        <node-number2|node-name2> <port-number2|port-name2>
```

A sample network topology description file is shown in Figure 5.14 on page 116.

```
# node switch# name
node 1 switch1
node 2 switch2
node 3 switch3
node 4 switch4
node 5 switch5
# port switch# port# port-name
port 1 1 PortA
port 1 2 PortB
port 2 1 PortA
port 2 2 PortB
port 2 3 PortC
port 3 1 PortA
port 3 2 PortB
port 3 3 PortC
port 3 4 PortD
port 4 1 PortA
port 4 2 PortB
port 5 1 PortA
port 5 2 PortB
port 5 3 PortC
# link switch# port# switch# port#
link 1 2 2 1
link 1 1 3 1
link 2 2 3 2
link 2 3 5 1
link 3 3 5 2
link 3 4 4 1
link 4 2 5 3
```

Figure 5.14 The sample network topology description file

The network described by this sample file is shown in Figure 5.15 on page 117. Note this is the same sample network depicted in Figure 3.1 on page 34. The difference is that here we only include switches, not other devices such as servers, etc.

Figure 5.15 The sample network

The output of the causal model generator would be 1666 lines, 480 of which are for vertices, and other 1186 lines are for edges. An excerpt of the causal model is shown in Figure on page 119. The syntax of the file is as follows:

- A vertex is specified by a line like the following:

```
vertex <vertex-number> <reserved> <generate-alarms> <root-cause> "<description>"
```

There are two values for the fields *root-cause* and *generate-alarms*: *true* or *false*.

For example, the following line specifies that there is a vertex 101, which is a root cause, but does not generate alarms, and whose description is "*Bit Error (switch1, PortA)*":

```
vertex 101 1 false true "Bit Error (switch1, PortA)"
```

- An edge is specified by a line which has the following syntax:

```
edge <source-vertex-number> <destination-vertex-number>  
<edge-number> <dependency-weight>
```

For example, the following line specifies that there is an edge from vertex 109 to 105, and the dependency weight is 7:

```
edge 109 105 36 7
```

```

vertex 35 1 true false "Correctable SAR_PDU Header Error (switch1)"
vertex 36 1 true false "Uncorrectable SAR_PDU Header Error (switch1)"
vertex 37 1 true false "Bad Sequence Number for AAL1 PDU (switch1)"
vertex 38 1 true false "Cells discarded in AAL1 SAR (switch1)"
edge 38 36 1 0.5
vertex 39 1 true false "SDT Error (switch1)"
edge 39 38 2 0.1
vertex 40 1 true false "Clock recovery fail (switch1)"
edge 40 38 3 0.5
vertex 41 1 true false "Cells arrives untimely (switch1)"
vertex 42 1 true false "Application glitches or performance degradation
(switch1)"
edge 42 38 4 0.2
(snip)
vertex 58 1 true false "Reassembly timeout for AAL5 PDU (switch1)"
vertex 59 1 true false "Assembly fails or retransmission for SSCOP PDU
(switch1)"
edge 59 57 28 0
edge 59 58 29 0
vertex 60 1 true false "Protocol anomaly/Performance (setup time, etc) degrada-
tion (switch1)"
edge 60 59 30 5
vertex 101 1 false true "Bit Error (switch1, PortA)"
vertex 102 1 false true "Physical Layer Protocol Error (switch1, PortA)"
vertex 103 1 false true "Link Down (switch1, PortA)"
vertex 104 1 false false "Cell Payload Bit Error (switch1, PortA)"
edge 104 101 31 2.14
vertex 105 1 false false "Cell Header Bit Error (switch1, PortA)"
edge 105 101 32 3.12
vertex 106 1 false false "Overhead Bit Error (switch1, PortA)"
edge 106 101 33 3.35
vertex 107 1 true false "BIP Error (switch1, PortA)"
edge 107 101 34 2.47712125471966243729
vertex 108 1 true false "Correctable HEC Error (switch1, PortA)"
edge 108 105 35 0.01
vertex 109 1 true false "Out of Cell Delineation (OCD) (switch1, PortA)"
edge 109 105 36 7
vertex 110 1 true false "Uncorrectable HEC Error (switch1, PortA)"
edge 110 105 37 3
vertex 111 1 true false "Alarms (LOS, LOF, etc.) (switch1, PortA)"
edge 111 102 38 0.6
edge 111 103 39 0.1
vertex 112 1 true false "Cells Dropped in Physical Layer (switch1, PortA)"
edge 112 109 40 0
edge 112 110 41 0
edge 112 111 42 0
(snip)

```

Figure 5.16 The sample topology specific causal model

Now suppose there is a fault called “*Traffic disagreement or wrong policing*” at *PortA* of *switch2*, then there will be numerous alarms generated. Table 5.4 on page 120 shows a list of alarms management application will receive:

Table 5.4 Alarms caused by the fault “Traffic disagreement or wrong policing” at PortA of switch2

Time	Vertex ID	Description of the Alarm
7.0	422	Cells discarded in ATM Layer (switch2, PortA)
7.3	222	Cells discarded in ATM Layer (switch1, PortB)
8.0	341	Cells arrives untimely (switch2)
8.6	337	Bad Sequence Number for AAL1 PDU (switch2)
8.8	339	SDT Error (switch2)
8.8	346	Bad Sequence Number for AAL3/4 PDU (switch2)
8.9	48	Reassembly time-out for AAL3/4 PDU (switch1)
8.9	347	Reassembly fails (wrong ST/SN, LI, CRC) (switch2)
10.4	348	Reassembly time-out for AAL3/4 PDU (switch2)
10.4	356	Length violation or CRC error (switch2)
10.5	357	Reassembly fails for AAL5 PDU (switch2)
10.6	358	Reassembly time-out for AAL5 PDU (switch2)

A sample alarm file for these alarms is shown in Figure 5.17 on page 121.

422
7.0
222
7.3
341
8.0
337
8.6
339
8.8
346
8.8
48
8.9
347
8.9
348
10.40
356
10.40
357
10.50
358
10.60

Figure 5.17 The sample alarm file

An excerpt of the output of the correlation program based on this alarm file is listed in Figure 5.18 on page 122. From the output we know that the correlation program finds the right cause. Also shown is that the correlation find the cause even there are missing alarms, and it also finds another cause (*"Traffic disagreement or wrong policing"* at *PortC* of *switch2*), but ranks the latter below the former because there are more missing alarms (9 to 6 for the former) for the latter.

The length bound is 23, and the weight bound is 3.0
There were 12 alarms generated.

Time	Vertex ID	Name of the Alarm
7.0	422	"Cells discarded in ATM Layer (switch2, PortA)"
7.3	222	"Cells discarded in ATM Layer (switch1, PortB)"
8.0	341	"Cells arrives untimely (switch2)"
8.6	337	"Bad Sequence Number for AAL1 PDU (switch2)"
8.8	339	"SDT Error (switch2)"
8.8	346	"Bad Sequence Number for AAL3/4 PDU (switch2)"
8.9	48	"Reassembly timeout for AAL3/4 PDU (switch1)"
8.9	347	"Reassembly fails (wrong ST/SN, LI, CRC) (switch2)"
10.4	348	"Reassembly timeout for AAL3/4 PDU (switch2)"
10.4	356	"Length violation or CRC error (switch2)"
10.5	357	"Reassembly fails for AAL5 PDU (switch2)"
10.6	358	"Reassembly timeout for AAL5 PDU (switch2)"

Cause 416 ("Traffic disagreement or wrong policing (switch2, PortA)")
explains 12 alarms.

alarm 48 ("Reassembly timeout for AAL3/4 PDU (switch1)")
Number of non-alarms in path = 2
Dependency weight of the path = 1.6010301

alarm 222 ("Cells discarded in ATM Layer (switch1, PortB)")
Number of non-alarms in path = 2
Dependency weight of the path = 2.5

alarm 337 ("Bad Sequence Number for AAL1 PDU (switch2)")
Number of non-alarms in path = 1
Dependency weight of the path = 2.9771214

(snip)

alarm 358 ("Reassembly timeout for AAL5 PDU (switch2)")
Number of non-alarms in path = 2
Dependency weight of the path = 1.4771214

alarm 422 ("Cells discarded in ATM Layer (switch2, PortA)")
Number of non-alarms in path = 1
Dependency weight of the path = 2.0

For this solution, the total number of non-alarms = 6
the total dependency weight = 27.695122

=====
Cause 616 ("Traffic disagreement or wrong policing (switch2, PortC)")
explains 12 alarms.

(snip)

For this solution, the total number of non-alarms = 9
the total dependency weight = 16.042728

Figure 5.18 The sample output of the alarm correlation program

5.2.3 Performance measurements

We measured the performance of the correlation algorithm from 3 to 5 switches. The important performance measurement results are shown in Table 5.5. The correlation time was measured on a Sun Ultra Sparc 10. Note that the numbers in the column "Time for resolution" are shown in a format $x+y$, where x stands for the time for reading in path repository and transitive closure construction (the non-real-time part), while y stands for the time for actual alarm correlation (the real-time part).

Table 5.5 Related numbers

Number of switches	Number of vertices	Number of edges	Number of paths	Time for repository (seconds)	Time for resolution (seconds)
3 (without bounds, i.e., Brunet's version)	178	350	8636	38	11+2
4 (with 5 as weight bound)	354	836	81702	9240	146+5
4 (with 4 as weight bound)	354	836	39556	1800	62+4
5 (without bounds, i.e., Brunet's version)	480	1186	>3,000,000	> 1,800,000 (500 hours)	not available
5 (with 4 as weight bound)	480	1186	84776	9480	149+4
5 (with 3 as weight bound)	480	1186	20568	184	30+3

Now we analyze the results and verify what we have got in Section 4.4 on page 78 from theoretical analysis:

- The experiments verified that the complexity of non-real time part is much higher than the real time part. For a complicated model, the non-real time part contributes to 95% of the time, this is shown in Table 5.5. The separation of the real time and non-real time calculation greatly improves the speed, since the non-real time calculation needs to be done only once per causal graph.
- It is also verified that with Brunet's algorithm the number of paths in the path repository grows very rapidly (it could grow exponentially as we said in Section 4.4.2 on page 79). As a result, it requires a lot of memory when the number of vertices and edges is large. Actually as shown in Table 5.5 on page 123, when there are 5 switches and no bounds are applied during path repository construction (Brunet's algorithm), the number of paths is so large that there was no enough memory to complete path repository construction. Thus we do not have performance data for this case.
- It is also verified that our performance improvements to Brunet's algorithm greatly improve the speed and memory requirements. As shown in the table, when there are 5 switches and the weight bound is 4, the non-real time part is about 100 times faster than that of Brunet's, while the memory requirements (number of paths) are just 3% of that of Brunet's.

Chapter 6 Conclusions

We conclude this thesis by briefly reviewing what we have done so far, by stating the contributions this work has brought, and by listing some work worth to be done in the future.

In Chapter 1, we reviewed the requirements of ATM network management and important issues, and the current status of alarm correlation algorithms and alarm correlation in ATM networks.

In Chapter 2, we reviewed the standards for ATM network management based on SNMP. Most of the alarms listed to be correlated in this thesis are based on MIB objects/traps defined in these standards.

In Chapter 3, we described the aspects of ATM network management and what we have done in SNMP-based ATM network management of several Nortel's ATM networks.

In Chapter 4, we reviewed two alarm correlation algorithms - Katzela's and Brunet's. We evaluated the scalability of Brunet's algorithm and found that it could have an exponential complexity. We then described what improvements we have done to Brunet's algorithm. We

showed that our improvements avoid an exponential complexity. We also compared the scalability of Katzela's algorithm, Brunet's algorithm, and our improvements.

In Chapter 5, we described the alarm propagation in ATM networks and illustrated the causal graph. We discussed how to extend Brunet's model and apply Brunet's algorithm to do alarm correlation in ATM networks. We described how to design a method to generate topology specific causal model automatically. We also measured the performance and scalability of Brunet's algorithm and our improvements.

6.1 Work done for this thesis

Now we summarize what we have done for this thesis:

- Studied and described how faults/alarms propagate in ATM networks. We described how faults/alarms propagate in a switch (including between ports in the same switch), and between two connected ports in two switches.
- Designed and developed a program called *Causal Model Generator* to generate a causal graph for an ATM network automatically, based on the network topology.
- We evaluated and improved the scalability of Brunet's algorithm. We have shown that Brunet's algorithm could have an exponential complexity and how we have improved it. We evaluated the complexity of both Brunet's algorithm and our improvements.
- Extended Brunet's model to address some shortcomings of Brunet's algorithm and applied Brunet's algorithm to do alarm correlation in ATM networks.

- We measured the performance of Brunet's algorithm and our improvements. The measurements verified our evaluation of the scalability of Brunet's algorithm and our improvements, we have shown that our improvements greatly improve the scalability.
- We also described how we managed some ATM LANs in Nortel.

6.2 Future work

With the work done for this thesis described above, the algorithm now has good accuracy and reasonable scalability. We also made it possible that a topology-specific causal graph be generated automatically so construction of a causal graph for a real ATM network is straightforward. So it is almost feasible to do alarm correlation for a real ATM networks. However, we still need to extend our causal model to include more faults/alarms there may be in the real ATM networks (like the alarms from proprietary MIBs) to make the causal model more closely reflect what faults a network management system (NMS) should be able to handle, and what alarms an NMS may get. We also still need to improve the scalability further to make it really useful in large ATM networks which may have already been made operational. In this section we suggest some work to be done to make the causal model more practical and to make the algorithm even more scalable.

There are some work worth to be done for causal model for ATM networks:

- Include more alarms from other layers. For example, it will be interesting to see how alarms from LANE (LAN emulation) can be incorporated to our causal model. Also as ATM networks get more and more widespread and more applications are running in ATM

networks, there will be other standards defined for these applications and by then we can incorporate alarms from those standards as well. It may be also desirable in practice to include alarms from proprietary MIB objects/traps.

- Make the estimates for probabilities more accurate. As we said in Chapter 5, it is very difficult to accurately estimate the probabilities of faults and causal relationships. Although Brunet's algorithm is not so sensitive to the probabilities as Katzela's algorithm, inaccurate probabilities could still lead to wrong or inaccurate diagnosis. As there is more experience in running ATM networks, it may be possible to get a much more accurate estimates of probabilities and make the diagnosis more accurate as well.

There are several possible ways to improve Brunet's algorithm (These have not been done because of time constraints):

- Count for aggregate probabilities. This would make it more accurate when there are multiple non-exclusive paths between a fault and an alarm.
- Remove the requirement that the whole path repository be read into memory to do correlation. Instead, read in only those paths originating from those vertices whose alarms have been received to be correlated. E.g., if we get two alarms 48 and 348, we need only read in paths originated from vertices 48 and 348.
- Use database to store the path repository, this would relieve memory requirements for large models.

- A way to store only frequently used paths in repository rather than the whole path repository. This is another way to relieve memory requirements for large models by compromising the speed.
- Compress the path repository: Currently when a path is stored in path repository, so are its sub-paths. This can be improved by only storing one path targeted at a cause, not saving the sub-paths targeted at the same cause.
- Allow probabilities for vertices (faults) to be specified. We have shown how to extend the model (by introducing another vertex) to circumvent the problem, however, we pay the price of more vertices and higher complexity. Incorporating these probabilities will spare ourselves from introducing additional vertices and higher complexity.

References

- [ATMF1994] ATM Forum Technical Committee, *Customer Network Management (CNM) for ATM Public Network Service (M3 Specification) af-nm-0019.000, Revision 1.04*, ATM Forum, October, 1994.
- [ATMF1996] ATM Forum Technical Committee, *Integrated Local Management Interface (ILMI) Specification, Version 4.0 (af-ilmi-0065.000)*, ATM Forum, September, 1996.
- [ATMF1997] ATM Forum Technical Committee, *Remote Monitoring MIB Extensions for ATM Networks (AF-NM-TEST-0080.000)*, ATM Forum, May, 1997.
- [ATMF1998a] ATM Forum Technical Committee, *M4 Interface Requirements and Logical MIB: ATM Network Element View (AF-NM-0020.001)*, ATM Forum, July, 1998.
- [ATMF1998b] ATM Forum Technical Committee, *SNMP M4 Network Element View MIB (AF-NM-0095.0014)*, ATM Forum, July, 1998.
- [Bouloutas1994] Bouloutas, A. T., Calo, S., Finkel, A., *Alarm Correlation and Fault Identification in Communication Networks*, IEEE Transaction on Communications, vol. 42, No. 2-4, pp. 523-533, February-April 1994.
- [Brunet1998] Brunet, R., *Temporal Alarm Correlation in communication networks*, Master of Science thesis, Carleton University, 1998.
- [IETF1996] Mouradiany, G., *Managed Objects for Recording ATM Performance History Data Based on 15 Minute Intervals*, IETF AToM MIB Working Group, March 1998.
- [IETF1998a] Ly, F., Noto, M., Smith A., Spiegel, E. M., Tesink, K., *Definitions of Supplemental Managed Objects for ATM Management*, IETF AToM MIB Working Group, March 1998.
- [IETF1998b] Noto, Tesink, K., *Definitions of Tests for ATM Management*, IETF AToM MIB Working Group, September 1998.

-
- [IETF1998c] McCloghrie, K., Heinanen, J., Greene, W., Prasad, A., *Managed Objects for Controlling the Collection and Storage of Accounting Information for Connection-Oriented Networks*, IETF AToM MIB Working Group, October 1998.
- [IETF1998d] McCloghrie, K., Heinanen, J., Greene, W., Prasad, A., *Accounting Information for ATM Networks*, IETF AToM MIB Working Group, October 1998.
- [IETF1998e] Noto, M., Spiegel, E. M., Tesink, K., *Definitions of Textual Conventions and OBJECT-IDENTITIES for ATM Management*, IETF AToM MIB Working Group, October 1998.
- [IETF1998f] Tesink, K., *Definitions of Managed Objects for ATM Management*, IETF AToM MIB Working Group, October 1998.
- [IETF1998g] Tesink, K., *Definitions of Managed Objects for the SONET/SDH Interface Type*, IETF AToM MIB Working Group, October 1998.
- [Katzela1995] Katzela, I., Schwartz, M., *Schemes for Fault Identification in Communication Networks*, IEEE/ACM Transaction on Networking, vol.3, No. 6, 1995.
- [Katzela1996] Katzela, I., *Fault Diagnosis in Communication Networks*, Ph.D. Thesis, Columbia University, 1996.
- [Lin1996] Lin, Y. D., Yang, R. K., Lo, C. T., *Alarm Correlation for congestion diagnosis in ATM Networks*, NOMS 96, 1996 IEEE Network Operations and Management Symposium, vol.2, pp. 624-627.
- [Minoli1996] Minoli, D., and Golway, T., *Planning and managing ATM Networks*, Manning.
- [RFC1595] Brown, T., Tesink, K., *Request for Comments: 1595, Definitions of Managed Objects for the SONET/SDH Interface Type*, IETF Network Working Group, March 1994.
- [RFC1695] Ahmed, M., Tesink, K., *Request for Comments: 1695, Definitions of Managed Objects for ATM Management Version 8.0 using SMIv2*, IETF Network Working Group, August 1994.
- [RFC2493] Tesink, K., *Request for Comments: 2493: Textual Conventions for MIB Modules Using Performance History Based on 15 Minute Interval*, IETF Network Working Group, January 1999.
- [RFC2512] McCloghrie, K., Heinanen, J., Greene, W., Prasad, A., *Request for Comments: 2512, Accounting Information for ATM Networks*, IETF Network Working Group, February 1999.

- [RFC2514] Noto, M., Spiegel, E., Tesink, K., *Request for Comments: 2514, Definitions of Textual Conventions and OBJECT-IDENTITIES for ATM Management*, IETF Network Working Group, February 1999.
- [RFC2515] Tesink, K., *Request for Comments: 2515, Definitions of Managed Objects for ATM Management Version 8.0 using SMIv2*, IETF Network Working Group, February 1999.
- [Yemini1994] Kliger, S., Yemini, S., Yemini, Y., Ohsie, D., Stolfo, S., *A Coding Approach to Event Correlation in Integrated Network Management IV*, Elsevier Science Publishers, 1994, pp. 266-277
- [Yemini1996] Yemini, S., Kliger, S., Mozes, E., Yemini, Y., Ohsie, D., *High-speed and Robust Event Correlation*, IEEE Communications Magazine, vol. 34, No. 5, pp. 82-90, May 1996.